LEARNING PATCH-BASED STRUCTURAL ELEMENT MODELS WITH HIERARCHICAL PALETTES

by

Jeroen Chua

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto

Abstract

Learning Patch-Based Structural Element Models With Hierarchical Palettes

Jeroen Chua
Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto
2012

Image patches can be factorized into 'shapelets' that describe segmentation patterns, and palettes that describe how to paint the segments. This allows a flexible factorization of local shape (segmentation patterns) and appearance (palettes), which we argue is useful for tasks such as object and scene recognition. Here, we introduce the 'shapelet' model- a framework that is able to learn a library of 'shapelet' segmentation patterns to capture local shape, and hierarchical palettes of colors to capture appearance. Using a learned shapelet library, image patches can be analyzed using a variational technique to produce descriptors that separately describe local shape and local appearance. These descriptors can be used for high-level vision tasks, such as object and scene recognition. We show that the shapelet model is competitive with SIFT-based methods and structure element (stel) model variants on the object recognition datasets Caltech28 and Caltech101, and the scene recognition dataset All-I-Have-Seen.

Acknowledgements

I'd like to acknowledge my parents for their general support, and Brendan Frey, Inmar Givoni, David Fleet, and Marcus Burbaker for their academic support. Finally, I'd like to acknowledge the University of Toronto board game club for fun times spent waiting for experiments to complete on Friday nights.

Contents

Chapter 1

Introduction

1.1 Motivation

Separating the description of an object's shape from appearance details, such as color and low-level texture, is an important problem in the field of computer vision [15]. Finding a factorized description of shape and appearance is especially useful for object recognition [10, 19, 20, 5, 9, 21]. Some object classes are delineated from other object classes by shape alone, others by appearance alone, and others by a mix of the two (see Figure 1.1), and so having a factorized description of shape and appearance aids in discovering salient aspects of an object.

The importance of factorizing descriptions of shape and color is especially relevant when examining common object recognition paradigms [24, 12, 5, 25]. For clarity of presentation, we will outline a common object recognition paradigm where it is assumed only one instance from one object is present in an image. Given an image $\vec{x}$, and C possible object classes, the goal of object recognition is to predict the object class contained within the image. This is a challenging task as objects can greatly vary in appearance due to a myriad of reasons, including occlusion, lighting variations, structural changes (*eg* a car without a roof), articulated pose, and viewpoint changes. In order to cope with such a wide range of variability in object appearance, object recognition is typically performed in successive stages, with each stage hopefully providing invariance to some of the previously described "nuisance" effects. The first stage involves defining or learning a function $f(\vec{x})$ with which to extract *image features*, $\vec{v}$, from the image. These image features are meant to capture high-order information about the depicted object, such as shape information, that is invariant to the aforementioned nuisance effects. Typical image features include image-gradient information [14, 4], phase responses [22], and responses to a set of basis functions that capture shape and texture information [18]. Given image features $\vec{v} = f(\vec{x})$, one then makes predictions using a classifier function, $g(\vec{v}) = y, y \in \{1...C\}$, which outputs a prediction for the object class present in the image. The classifier stage serves as a catch-all for nuisance effects not explicitly dealt with by the image features. For example, if viewpoint invariance is a desirable property of an object recognition framework, but the extracted image features are sensitive to viewpoint, then it is left to the classifier to make the entire system viewpoint invariant. Popular classifier functions include SVMs, K-nearest-neighbors, and multinomial logistic regression. The described object recognition paradigm is shown in Figure 1.2.

Examining the framework shown in Figure 1.2, we note that object recognition performance is sensitive to the kinds of image features, $\vec{v}$ given to the classifier, and much work has gone into what constitutes a "good" image features [14, 4, 22, 18, 17]. Intuitively, a good image feature extraction function makes the role of the classifier trivial by having all desired invariance properties, and extracting features such that object classes are

Figure 1.1: Examples of object classes in which shape and appearance descriptions have differing discriminative powers. For example, cars (left) are primarily characterized by their shape and constituent parts, while appearance features such as color play a lesser role. On the other hand, apple juice (middle) is primarily distinguished by its appearance features, such as the specific shade of golden brown and its perceived transparency, while the shape it takes is largely irrelevant. Finally, a sunflower (right) is characterized by both shape and color.

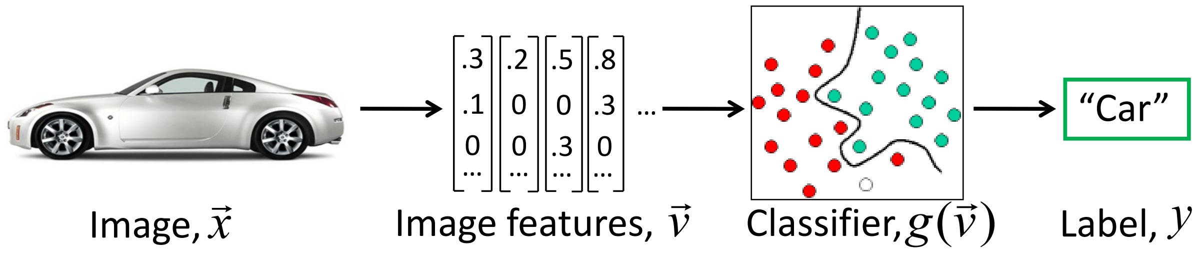

Figure 1.2: A framework for doing object recognition. Starting from an image $\vec{x}$, image features $\vec{v}$ are extracted, which are then fed into a classifier function, $g(\vec{v})$, which outputs an object label (in this case, "Car").

easily delineated. Recall from Figure 1.1 that object classes vary in how discriminative shape and appearance descriptions are, and so it is desirable to extract a set of features that is invariant to shape changes (if it is appearance that is important) and a set that is invariant to color changes (if it is shape that is important). In light of this, it is reasonable to suggest constructing two kinds of image features- one that describes shape alone, and one that described appearance alone. This way, a classifier can learn which source of information is most discriminative, or more generally, how to combine the evidence from shape and appearance features. Additionally, the classifier is not saddled with the difficult task of gaining shape and appearance invariance. We argue that operating in such a $\mathrm{shape} \times \mathrm{appearance}$ image feature space will ease the burden of the classifier by making it more apparent how to distinguish once class from another. One of the primary motivations of this work is to find "good" factorized descriptions of shape and appearance.

As will be discussed in Section 1.2, there has been much research on factorizing shape and color for object recognition [10, 19, 20]. However, such approaches typically rely either on training and test images being aligned (*ie* objects are in the same part of the image, and same scale and orientation) or that it is possible to compute image alignments to a canonical reference frame [10, 20]. Computing the optimal image alignment often cannot be found exactly and requires either an expensive search over transformation parameters (*eg*, translations), or using unrealistically aligned datasets to sidestep the need to infer transformation parameters altogether. Unfortunately, many kinds of objects naturally undergo extreme variations in deformation and articulation (*eg*, a person in various poses), and so it is necessary to perform the search over transformation parameters. This search over

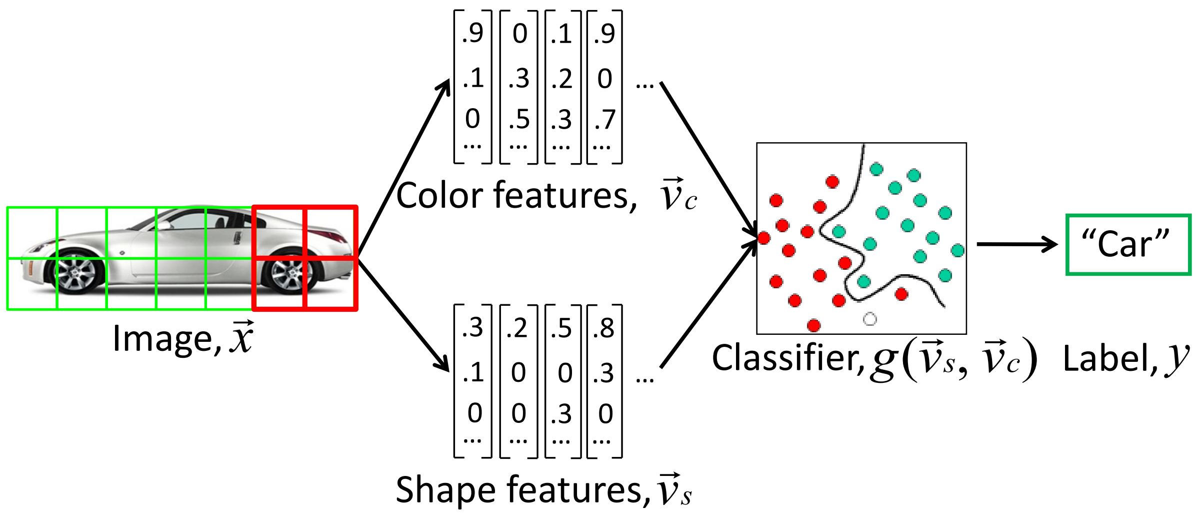

Figure 1.3: Our object recognition framework. We operate on patches to extract shape and color descriptors separately. Note that we extract one shape descriptor and one color descriptor per patch (illustrated by the four red squares, and the four shape and color descriptors). We argue this approach combines the flexibility of patch-based models with an efficient image representation afforded by factorizing shape and color descriptions.

transformation parameters is typically phrased as an optimization problem, and a wide class of algorithms use the Expectation-Maximization (EM) algorithm to perform this optimization. However, EM in this case is extremely prone to falling into local minima, and so the transformation parameter estimates tend to be poor. In addition, such models have difficulty reasoning about occlusions as they presume that the entire object is visible, and it is difficult to outfit them to handle such situations. These shortcoming limit the practical applications of existing approaches that factorize shape and color [10, 19, 20, 9].

Recently, patch-based approaches [24, 12] have seen great success on object recognition tasks and seem able to cope with a large range of object deformations. Here, (patch) features are extracted in a dense tiling over spatial regions of an image to form an image feature representation. Such approaches overcome the need to infer transformations by relying on statistics of *local* image features to perform recognition, thus foregoing the requirement that entire images be well-aligned. Also, because local image features are used, patch-based approaches can handle some occlusion as long as the local image statistics do not suffer extreme degradation (empheg, due to a large degree of occlusion). However, one drawback with current patch-based approaches is that they typically do not factorize shape and appearance and as we have previously discussed, such factorizations are desirable.

In this work, we propose a patch-based framework that factorizes shape and appearance. Our approach operates in a patch-based manner, extracting shape and color descriptors for each patch. We then use these patch-based descriptors for high-level vision tasks, such as object and scene recognition. A sketch of the framework we propose is outlined in Figure 1.3, and is described in more detail in Chapter 2. Our approach combines the flexibility of patch-based models with an efficient image representation afforded by factorizing shape and color descriptions.

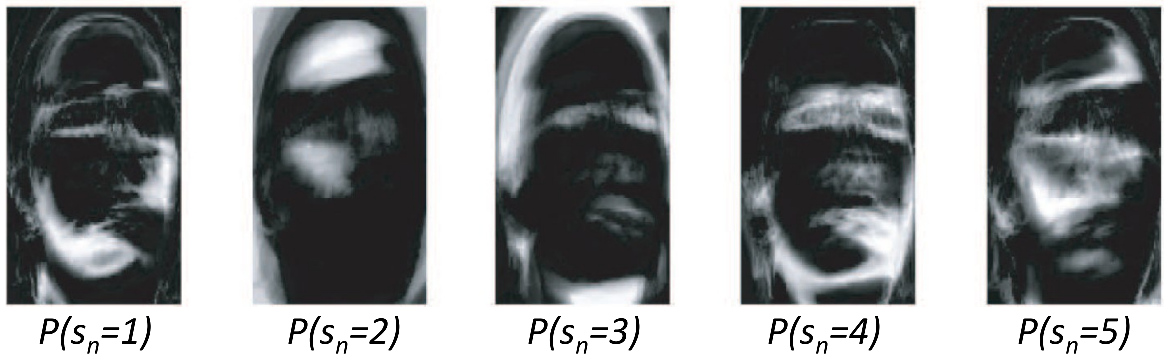

Figure 1.4: Example of structure learned by the PIM model for "faces" for $S = 5$ index maps. Note how pixels corresponding to regions that typical co-occur in color, such as forehead and cheek pixels ($s = 2$), and hair pixels ($s = 3$) are grouped together. Visualization shown is of $P(s_n = s)$, where white pixels indicate high probability of assignment. PIM maps are taken from [10].

1.2 Background

In this section, we describe relevant previous approaches to factorizing shape and appearance, as well as a state-of-the-art patch-based framework that does not separate shape and appearance in its image feature representation.

1.2.1 Probabilistic Index Maps

Our approach is highly influenced by the Probabilistic Index Maps (PIM) method [10], and this is the most related work to our proposed model. The PIM method is also referred to as the "structure element model" (stel model). Here, we describe the core idea behind PIMs, as introduced in [10]. Although there are many variants on PIMs described in [10], we describe only the basic idea as that is the most relevant for our discussion.

The premise behind the PIM model is that while object classes may undergo large appearance changes (*eg*, under different illuminations and viewpoints), the shape (structural) properties behind object instances are largely the same. For example, a person wearing a red shirt or a green shirt may, on a pixel-color level, look vastly different, but are shaped in a similar kind of way. This model then seeks to model the shape of a class of objects independently of appearance.

Suppose we have M images of an object class, with each image containing N pixels. The PIM model captures the structure of an object class by way of a probabilistic index map which associates a discrete distribution over S indices for each of the N pixels. Throughout the image, each of the S indices represents a probabilistic grouping of pixels that tend to co-occur in color over all training images of a given object class. For example, the $s - th$ group might represent a group of pixels corresponding to the forehead of a person. We denote the probability that a pixel n is assigned the index s as $P(s_n = s)$. Note that all M images of an object class share the probabilistic index map. An example of a learned PIM model for faces is shown in Figure 1.4, and is taken from [10]. Note that the probabilistic index map specifies which pixels tend to co-occur in appearance without specifying what appearance that should be.

We have described how structure is modelled in the PIM model, and now address how appearance is modelled. The PIM model assumes that while the structure of an object class may be the same across images, instances of the class may differ in appearance from image to image. To deal with this appearance variability, the model defines, for *each* image m, a set of S palettes which specify the appearance that pixels belonging to the $s-th$ group takes on in that image. One example of possible palette parameters are a mean color, $\vec{\mu}_s^m$ and a covariance matrix $\mathbf{\Sigma}_s^m$ for each of the S image palettes. In this case, it is assumed that all pixels in image m assigned to index s are well-approximated by the color $\vec{\mu}_s^m$, with some covariance $\mathbf{\Sigma_s^m}$. This way, images that have similar structure but vastly different appearances can be described by similar probabilistic index map assignments, $s_n, n \in [1...N]$, but different palettes $\{\vec{\mu}_s^m, \mathbf{\Sigma}_s^m, s \in [1...S]\}$. Note the separation of specification of structure from the specification of appearance. This allows the PIM model to disentangle shape (in the form of the probabilistic index maps) from appearance (in the form of palettes), and serves as a basis for the work we propose.

We stress that in this approach, entire object classes are modelled with a single probabilistic index map, and it is implicitly assumed that images can be roughly aligned. As we have argued before, this makes the method sensitive to the requirement of image alignment, and unable to be robust to object occlusions.

1.2.2 SIFT + Vector Quantization

Patch-based approaches for object recognition have recently become popular. Chief among these approaches is the use of dense Scale-Invariant Feature Transforms (SIFT) features [14], coupled with vector quantization. In this section, we outline the operation of this approach. Our goal here is to sketch out the high-level operation of the SIFT feature extraction stage, and the vector quantization stage. There are many small details of the SIFT extraction stage that are not critical to motivate or understand this work, such as the Gaussian weighting window used to weight pixel gradients and the way in which the resulting feature vector is normalized, and so we will not discuss these details to keep our treatment as simple as possible.

SIFT features [14], extract statistics relating to image gradients in image patches and have been used in a wide array of applications including object recognition [24, 12], image stitching [1], and video tracking [27]. While a classic use of SIFT features has been feature matching, here we describe its use in an object recognition framework. This will simplify our discussion since a large amount of machinery in extracting SIFT descriptors as described in [14] is not performed in the object recognition setting. In particular, we will forego descriptions of the *interest point* selection method for SIFT features, and the machinery involved in approximately matching SIFT features across two images.

In an object recognition framework, SIFT features are first extracted from a set of patches (typically overlapping) all over an image. Patches are typically 16×16 pixels, and are spaced so that their centres are only 4 pixels apart. To extract a SIFT feature for a given 16×16 pixel patch, one first divides the patch into 16 non-overlapping blocks of 4×4 pixels. Next, statistics relating to pixel gradients within each of the 16 blocks are computed and are binned into eight bins, according to each gradient's orientation. This results in an 8-dimensional descriptor for each of the 16 blocks. These descriptors are then stacked together to form a patch descriptor that is 128-dimensional. A pictorial illustration of the (simplified) SIFT extraction method is shown in Figure 1.5. Note that since the SIFT features depends on statistics of pixel gradients, it contains a representation of local shape in a patch intertwined with a representation of local coloring. For example, SIFT descriptors for image patches with two vertical stripes of colors will, in general, look very different; a patch with red on the left and green on the right will have a different SIFT descriptor than a patch with red on the left and purple on the right. This is highly undesirable if appearance invariance is required; that is, if what is important is that there is two vertical stripes in the patch, and not so much the exact colors of the stripes. In other words, SIFT features do *not* factorize shape and color in its representation.

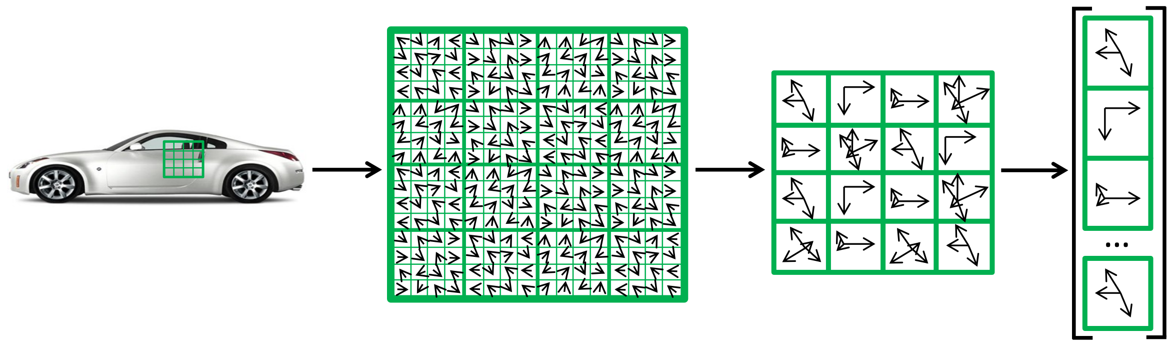

Figure 1.5: A simplified pictorial illustration of SIFT feature construction. First, a 16×16 patch of pixels is extracted from the image and divided into 16 blocks of 4×4 pixels. Then, statistics relating to image gradients are computed at each pixel and binned into one of eight bins, depending on orientation. After, the binned image gradients are accumulated within each of the 16 blocks. Finally, the binned image gradients are stacked into a vector, yielding a 128-dimensional feature vector describing the image patch. Arrows within the the green grids are used to indicate (example) directions and magnitudes of the image gradients located in the delineated subregion. This process of SIFT feature extraction is repeated over a dense, possibly overlapping set of 16×16 pixel patches around the image.

Vector quantization

After SIFT features have been extracted from a training set of images, the SIFT features are quantized (clustered) into a set of K clusters. In computer vision literature, each of the K clusters are also referred to as codewords, and the set of clusters referred to as a codebook. The clustering object function is typically taken to be the K-means clustering objective function, and clustering is performed using the K-means algorithm [12, 24]. Following this, SIFT features are then hard-assigned to its nearest cluster centre.

Clustering the SIFT features has a number of advantages, including gaining robustness to a small amount of unstructured noise, and making the feature representation much more compact. Rather than having 128 real numbers used to represent a feature, $ceil(\log(K))$ bits can be used instead, where $ceil()$ is the ceiling function. This results in potentially massive savings in memory footprint, which is crucial for large-scale applications.

Chapter 2

Model

In this section, we describe our patch-based approach to factorizing shape and color. We note that our model is different than patch-based approaches, such as those based on SIFT descriptors described in Section 1.2, since we explicitly factorize and color. Also, our model is different than other stel model variants since we operate in a patch-based framework.

Given an unlabeled set of training images, the shapelet model attempts to explain all image patches using a library of patch shapes, or shapelets, and a hierarchical set of palettes used to color each patch in each image. The shapelets capture information concerning local shape in a patch regardless of the specific colors used, whereas the palettes capture information concerning the coloring of patches and images regardless of the patch or image structure. A hierarchical palette of colors is used to capture coloring at the patch level and image level. In addition, a universal palette accounts for the colors of all training images. Each image selects a subset of image colors from the universal palette to form the image palette, and in turn, each patch in the image chooses a subset of colors from the image palette to form the patch palette. The hierarchy of palettes enables the model to capture inter and intra-image coloring consistencies.

Using a generative modeling approach, a shapelet is a probabilistic grouping of pixels in a patch into a set of regions that reflect co-occurrence of color. Shapelets capture local image structure by modeling this co-occurrence without regard for the specific color identities. In order to explain patches of differing complexity, shapelets are allowed to contain a differing number of regions. A shapelet with a single region describes a uniformly colored patch, while a shapelet with several regions can capture more complex patterns. An image patch can be described by a shapelet and a palette reflecting the color means and variances of each shapelet region. This gives rise to a flexible patch-based image representation: multiple patches can be compactly explained by a single shapelet with different palettes. The overall idea of this approach is shown in Fig. 2.1.

Regions within a shapelet serve a similar role to stels [10, 19, 20, 9] in that both represent probabilistic pixel groupings based on color co-occurrence. However, stel models are often learned on whole images, and so stels often correspond to object parts (*eg*, the shirt of a person), while shapelets are learned on the patch level and correspond to shape primitives (*eg*, a quarter circle).

Given a test image and a learned library of shapelets, (approximate) posterior distributions describing which shapelets best model each patch in the image are inferred, along with the palettes used to color the patches and the image. This decomposed representation can be used to perform higher level vision tasks, such as object and scene recognition. The rest of this section provides technical details of the generative model and the learning algorithm employed in this work.

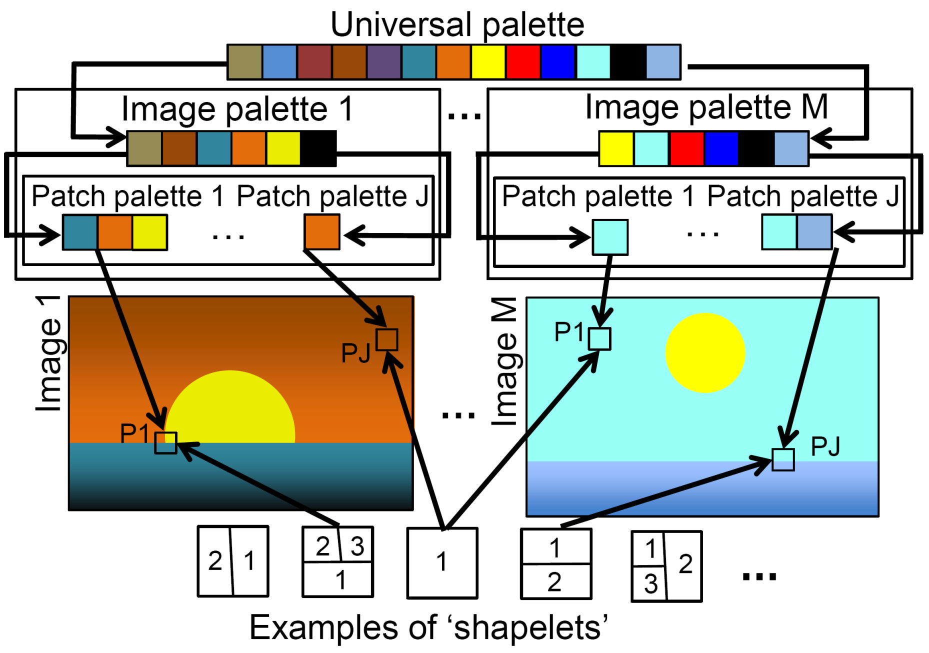

Figure 2.1: An illustration of our approach. Shapes comprising image patches can be described using 'shapelets' with varying numbers of regions (bottom of figure). To describe an image as a collection of shapelets, each shapelet region needs to be painted, or colored. From a universal palette (top of figure), each image selects a subset of image colors (top-middle), and from these image colors, each patch selects a subset of patch color (bottom-middle). Separately, for each patch, a shapelet from the shapelet library is chosen (bottom). The shapelet is then colored according to the chosen patch palette. In this way, our model explicitly factorizes local shape, in the form of shapelets, from appearance, in the form of hierarchical palettes.

2.1 Probabilistic graphical model for shapelets

Given M same-size images, each containing J non-overlapping patches of size $N_y \times N_x = N$ pixels, our goal is to learn a library of shapelets and to infer the universal, image, and patch palettes. Let S denote the number of shapelets in the library, U denote the number of colors in the universal palette, and G denote the number of colors in the image palettes. Also, let each shapelet s contain R_s regions, and let the patch palettes contain up to R colors, where $R = \max_s R_s$.

A shapelet s, containing R_s regions, is represented as a collection of N discrete distributions over the R_s region indices. For each shapelet, these distributions indicate the region preference of every pixel in a patch.

To generate a set of images from our model, one first generates a set of universal colors comprised of U means in some color space (*eg*, RGB). Next, to generate an image, an image palette is generated by randomly selecting G color means from the universal palette and specifying their variances (therefore color variances are image-specific). Allowing for image-specific variances for the image palette allows us to deal with data sources of varying noise levels and conditions (*eg*, cameras using different exposure settings). Then, each patch in the image picks one shapelet from the library of S shapelets. Each of the N patch pixels are assigned to one of the R_s regions by independently sampling from each pixel's discrete distribution, as defined by the selected shapelet. Finally, each of the R_s regions is assigned to one of the G image colors by sampling from their respective distributions defined by the patch palette, and uses their image color's mean and variance to color their assigned pixels. Note that multiple regions may pick the same image color; all that is required is that pixels belonging to the same region are explained by the same global color.

Therefore, in this model, the generation of local shape (specified by selecting a shapelet) is explicitly separated from the generation of local appearance (palettes); it is this separation that allows the shapelet model to efficiently factorize shape and appearance.

The graphical model is given in Fig. 2.2. $u_{mg} \in \{1...U\}$ is the universal color index being used in image m by image color g; the shapelet index being used in image m and patch j is $s_{mj} \in \{1...S\}$; the region index to which pixel n in image m and patch j, using shapelet s, is assigned is $r_{mjn} \in \{1...R_s\}$; the image color index to which pixel n in image m and patch j is assigned is $g_{mjn} \in \{1...G\}$; the observed value of pixel n in image m and patch j is denoted by $\vec{x}_{mjn} \in \mathbb{R}^H$, where H is the number of color channels (*eg*, $H = 3$ in the case of RGB-space).

The above hidden variable distributions are paramatrized as follows:

$$P(u_{mg} \,|\, \vec{\beta}) = \text{Discrete}(\vec{\beta}), \vec{\beta} \in \mathcal{R}^U \tag{2.1}$$

$$P(s_{mj} \,|\, \vec{\theta}) = \text{Discrete}(\vec{\theta}), \vec{\theta} \in \mathcal{R}^S \tag{2.2}$$

$$P(r_{mjn} \,|\, s_{mj}{=}s, \vec{\pi}_{ns}) = \text{Discrete}(\vec{\pi}_{ns}), \vec{\pi}_{ns} \in \mathcal{R}^{R_s} \tag{2.3}$$

$$P(g_{mjn} \,|\, r_{mjn}{=}r, \vec{\alpha}_{mjr}) = \text{Discrete}(\vec{\alpha}_{mjr}), \vec{\alpha}_{mjr} \in \mathcal{R}^G \tag{2.4}$$

The parameter $\vec{\beta}$ controls the selection of colors from the universal palette that form image colors. The parameter $\vec{\theta}$ controls the prior over shapelets for modeling the local shape of a patch, and is further discussed below. $\vec{\pi}_{ns}$ parameterizes, for a pixel n in shapelet s, the distribution over the R_s regions and is analogous to a distribution over stel assignments [10, 19, 20, 9] for patches. Similarly, $\vec{\alpha}_{mjr}$ parameterizes, for a patch j in an image m for the region r, the distribution over the G image colors, and forms the lowest level of our color hierarchy (patch palette), as shown in Fig. 2.1.

Recall that our framework is capable of handling shapelets with a differing number of regions through an appropriate choice of the $R_s \forall s$. The choice of shapelet, and thus of shapelet complexity, is controlled by the parameter $\vec{\theta} = [\theta_1, \ldots, \theta_S]^T$. It is often desirable to either limit or regularize model complexity to obtain good generalization, and so the framework can be biased to prefer using shapelets with fewer regions ('simpler' shapelets). This is achieved by setting

$$\theta_s = \frac{\exp(-\lambda(R_s - 1)^2)}{\sum_{s=1}^{S} \exp(-\lambda(R_s - 1)^2)}, \tag{2.5}$$

where λ is a regularization parameter. This prior is fixed since R_s is fixed $\forall s$. Alternative forms of regularization are also possible, such as using a Dirichlet prior with S parameters. However, such a prior may require careful tuning to balance the use of simpler and more complex shapelets. In contrast, the regularization choice in Eq. 2.5 has only one tunable parameter, and in practice has the desired effect of encouraging the use of shapelets with fewer regions.

For the observation model, it is assumed that each H-dimensional pixel value is distributed according to an H-dimensional axis-aligned Gaussian:

$$P(\vec{x}_{mjn} \,|\, g_{mjn}{=}g, u_{mg}{=}u, \vec{\mu}_u, \vec{(\sigma^2)}_{mg}) = \prod_{h=1}^{H} \mathcal{N}(x^h_{mjn} \,|\, \mu^h_u, (\sigma^2)^h_{mg}), \tag{2.6}$$

where $\mathcal{N}()$ is the normal distribution, μ_u^h is the mean of color channel h of universal color u, and $(\sigma^2)_{mg}^h$ is the variance of color channel h of image color g in image m.

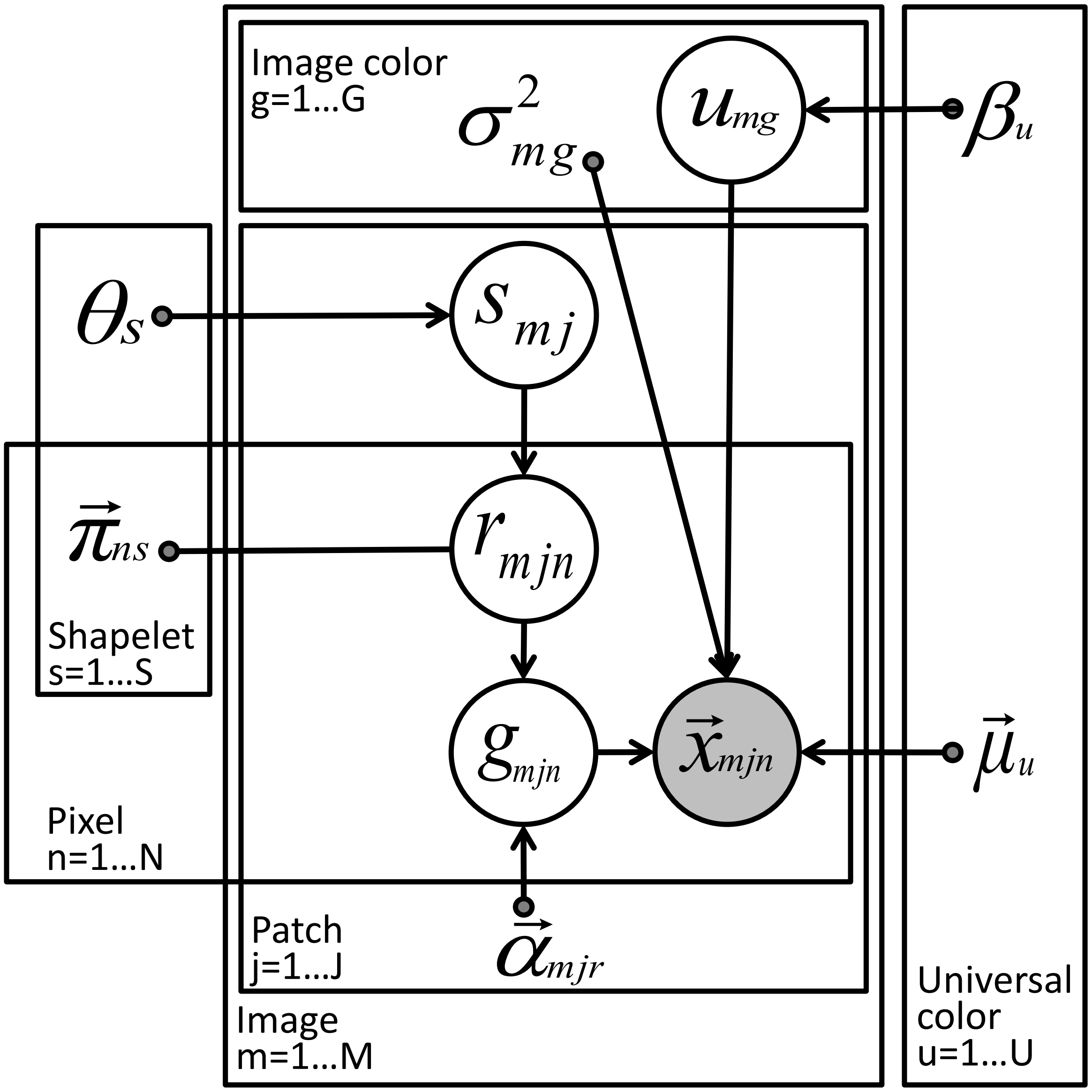

Figure 2.2: Graphical model for the shapelet model. An image m is generated by first selecting G image colors from a library of U universal colors. Then, each patch j in the image selects a shapelet, s_{mj}, to model its local shape, and $R_{s_{mj}}$ colors from the G image colors to model its local color. Each patch $\vec{x}_{mj}$ is then colored according to the choice of shapelet and palettes.

2.2 Variational learning

Here, we detail an algorithm to perform learning in the framework described in Section 2.1. For notational convenience, let $\Phi = \{\pi, \alpha, \beta, \mu, \sigma\}$ (the collection of model parameters), and $\mathbf{Z} = \{\mathbf{s}, \mathbf{r}, \mathbf{g}, \mathbf{u}\}$ (the collection of all hidden variables).

The goal of learning is find the model parameters, Φ, that maximize the data likelihood, $P(\mathbf{X} \,|\, \vec{\theta}, \Phi)$, or equivalently, the data log-likelihood, $\log P(\mathbf{X} \,|\, \vec{\theta}, \Phi)$. However, we are unable to directly optimize the data log-likelihood since our model contains a potentially large number of hidden variables. Instead, we work with the *complete* data log-likelihood, $\log P(\mathbf{X}, \mathbf{Z} \,|\, \vec{\theta}, \Phi)$, given by:

$$\begin{aligned}
&\log P(\mathbf{X}, \mathbf{Z} \,|\, \vec{\theta}, \Phi) = \\
&\sum_{mg} \log P(u_{mg} \,|\, \vec{\beta}) + \sum_{mj} \log P(s_{mj} \,|\, \theta_s) \\
&+ \sum_{mjn} \log P(r_{mjn} \,|\, s_{mj}, \vec{\pi}_{ns}) + \sum_{mjn} \log P(g_{mjn} \,|\, r_{mjn}, \vec{\alpha}_{mjr}) \\
&+ \sum_{mjn} \log P(\vec{x}_{mjn} \,|\, g_{mjn} = g, u_{mg} = u, \vec{\mu}_u, (\vec{\sigma^2})_{mg}).
\end{aligned} \tag{2.7}$$

Unfortunately, the global optimal setting of Φ cannot be found in closed-form. As a result, we employ a variational EM algorithm to perform learning. In general, there are three ingredients that must be specified or computed to execute a variational EM algorithm:

- A distribution over the hidden variables, $Q(\mathbf{Z})$ (Q-distribution)
- Expectation-step updates (E-step)
- Maximization-step updates (M-step)

We specify each of these ingredients in the remainder of this section.

2.2.1 Q-distributions

Here, we define the form of the Q-distribution, $Q(\mathbf{Z})$, that we use for learning in the variational EM framework. To keep the notation uncluttered, unless otherwise noted, we omit conditioning on the parameters $\mathbf{\Phi}$ and $\vec{\theta}$ for the remainder of this section.

The EM algorithm requires the specification of a probability distribution over hidden variables, $Q(\mathbf{Z})$. While we are free to specify the functional form of $Q(\mathbf{Z})$, a natural choice is the one that minimizes the free-energy F between $Q(\mathbf{Z})$ and $P(\mathbf{X}, \mathbf{Z})$:

$$F = \sum_{\mathbf{Z}} Q(\mathbf{Z}) \log \frac{Q(\mathbf{Z})}{P(\mathbf{X}, \mathbf{Z})}. \tag{2.8}$$

This is a natural objective to minimize as F is an upper-bound on the negative data log-likelihood, $-\log P(\mathbf{X})$, and so minimizing Eq. 2.8 is equivalent to maximizing a lower bound on the data log-likelihood [16]. Note also that $F = KL\Big(Q(\mathbf{Z}) \,||\, P(\mathbf{Z} \,|\, \mathbf{X})\Big) - \log P(\mathbf{X})$, where $KL(\cdot)$ is the Kullback-Leibler divergence. So, $F \geq -\log P(\mathbf{X})$ since $KL(\cdot) \geq 0$, and is minimized when $Q(\mathbf{Z}) = P(\mathbf{Z} \,|\, \mathbf{X})$. In other words, Eq. 2.8 is minimized when $Q(\mathbf{Z})$ is set to be the posterior distribution over the hidden variables.

Ideally, one would set $Q(\mathbf{Z}) = P(\mathbf{Z} \,|\, \mathbf{X})$ to minimize Eq. 2.8; however, this does not always lead to a form of $Q(\mathbf{Z})$ that is tractable to compute. In general, a complete description of the posterior distribution $P(\mathbf{Z} \,|\, \mathbf{X})$ where the hidden variables $\mathbf{Z}$ take on discrete values requires a table of $\prod_k^K |z_k| - 1$ values to be computed, where there are K total (discrete) hidden variables, and $|z_k|$ indicates the number of values the $k-th$ hidden variable may take. Since the number of hidden variables generally increases with the amount of data available and model complexity, it may be intractable to compute the true posterior distribution over discrete hidden variables. Because of this, setting $Q(\mathbf{Z}) = P(\mathbf{Z} \,|\, \mathbf{X})$ to minimize Eq. 2.8 is not always possible. Instead, one may consider a restricted form of $Q(\mathbf{Z})$ that is tractable to compute, yet is a good approximation to $P(\mathbf{Z} \,|\, \mathbf{X})$. We will see that the true posterior distribution $P(\mathbf{Z} \,|\, \mathbf{X})$ for the shapelet model is indeed intractable and we must restrict the form of $Q(\mathbf{Z})$ to one that is computationally feasible to work with.

Using the graphical model in Fig.2.2 as reference, we note that the following decomposition holds:

$$P(\mathbf{Z} \,|\, \mathbf{X}) = \prod_m P(\mathbf{Z}_m \,|\, \mathbf{X}_m) \tag{2.9}$$

where $\mathbf{Z}_m$ is the set of hidden variable relevant to image m, and $\mathbf{X}_m$ is the data observations for image m. Unfortunately, $P(\mathbf{Z}_m \,|\, \mathbf{X}_m)$ cannot be further decomposed, and is intractable since each image may have many associated hidden variables. As such, we cannot set $Q(\mathbf{Z}_m) = P(\mathbf{Z}_m \,|\, \mathbf{X}_m)$. Instead, we seek to restrict $Q(\mathbf{Z}_m)$ to a form that is tractable to work with, and is as faithful to $P(\mathbf{Z}_m \,|\, \mathbf{X}_m)$ as possible.

The approach we will employ is to find a *decomposed* form of $Q(\mathbf{Z}_m)$ since such forms lead to compact representations that can be computed efficiently. Since there may be many patches in a given image, it is desirable for $Q(\mathbf{Z}_m)$ to have a decomposed form over each patch, j. We note that even conditioned on the training image data, $\mathbf{X}_m$, and noting that all patches must draw from a common set of image colors, $P(\mathbf{Z}_m \,|\, \mathbf{X}_m)$ does **not** factorize over patches due to explaining away. Because of this, we use the following approximation for $Q(\mathbf{Z}_m)$:

$$P(\mathbf{Z}_m) \approx Q(\mathbf{s}_m, \mathbf{r}_m, \mathbf{g}_m, \mathbf{u}_m) = Q(\mathbf{s}_m, \mathbf{r}_m, \mathbf{g}_m) Q(\mathbf{u}_m) \tag{2.10}$$

where $\mathbf{u}_m \in \mathcal{R}^{G \times U}$ denotes the set of hidden variables that govern to which of the U universal colors each of the G image colors is assigned. As stated above, it is computationally desirable to restrict the Q-distributions to decomposed forms, and since G may be large, we seek a further decomposed form of $Q(\mathbf{u}_m)$. Examining the graphical model in Figure 2.2, $Q(\mathbf{u}_m)$ does not have a further decomposable form due to explaining away. Since $Q(u_{mg}) \forall g \in \{1...G\}$ is not independent conditioned on observing the image data, $\mathbf{X}_m$

We now turn to finding a decomposed form for $Q(\mathbf{s}_m, \mathbf{r}_m, \mathbf{g}_m)$. We employ the following approximation:

$$Q(\mathbf{s}_m, \mathbf{r}_m, \mathbf{g}_m) = \prod_j Q(s_{mj}) \prod_n Q(r_{mjn} \,|\, s_{mj}) Q(g_{mjn} \,|\, r_{mjn}). \tag{2.11}$$

This is an approximation since $Q(\mathbf{s}_m, \mathbf{r}_m, \mathbf{g}_m)$ does not factorize over patches, j, nor pixels, n, due to the sharing of the set of image colors.

Using the above approximations, we may now fully specify the Q-distributions used in our variational EM framework:

$$P(\mathbf{Z} \,|\, \mathbf{X}) = P(\mathbf{s}, \mathbf{r}, \mathbf{g}, \mathbf{u} \,|\, \mathbf{X}) \approx Q(\mathbf{Z}) = Q(\mathbf{s}, \mathbf{r}, \mathbf{g}, \mathbf{u}) \tag{2.12}$$

$$Q(\mathbf{s}, \mathbf{r}, \mathbf{g}, \mathbf{u}) = \prod_m \Big(\prod_g Q(u_{mg}) \Big) \prod_j Q(s_{mj}) \prod_n Q(r_{mjn} \,|\, s_{mj}) Q(g_{mjn} \,|\, r_{mjn}) \tag{2.13}$$

We have now defined a restricted, decomposed form of $Q(\mathbf{Z})$ that is tractable. Next, we describe how, given an estimate of the model parameters, $\mathbf{\Phi}$, we can minimize Eqn. 2.8 using the Q-distribution defined by Eqn. 2.11.

2.2.2 E-step

In this section, we define the algorithm to update the Q-distributions $Q(s_{mj})$, $Q(u_{mg})$, $Q(r_{mjn} \,|\, s_{mj})$, and $Q(g_{mjn} \,|\, r_{mjn})$ in order to minimize Eqn. 2.8, given an estimate of the model parameters, Φ. In the language of EM, this section outlines the E-step.

First, we explicitly substitute Eqn. 2.7 into Eqn. 2.8:

$$\begin{aligned} F = &\sum_{\mathbf{Z}} Q(\mathbf{Z}) \log Q(\mathbf{Z}) \\ &- \sum_{\mathbf{Z}} Q(\mathbf{Z}) \sum_{mg} \log P(u_{mg}) - \sum_{\mathbf{Z}} Q(\mathbf{Z}) \sum_{mj} \log P(s_{mj}) \\ &- \sum_{\mathbf{Z}} Q(\mathbf{Z}) \sum_{mjn} \log P(r_{mjn} \,|\, s_{mj}) - \sum_{\mathbf{Z}} Q(\mathbf{Z}) \sum_{mjn} \log P(g_{mjn} \,|\, r_{mjn}) \\ &- \sum_{\mathbf{Z}} Q(\mathbf{Z}) \sum_{mjn} \log P(\vec{x}_{mjn} \,|\, g_{mjn}{=}g, u_{mg}). \end{aligned} \tag{2.14}$$

We first consider the factor $Q(s_{mj})$. We seek the expression for $Q(s_{mj})$ that minimizes Eqn. 2.8, keeping all other Q-distributions fixed. We will refer to this expression as $Q^*(s_{mj})$. Taking the functional derivative of Eqn. 2.14 with respect to $Q(s_{mj})$ and setting it to zero yields

$$\begin{aligned} 0 = &\log Q^*(s_{mj}) && (2.15) \\ &- \log P(s_{mj}) \\ &- (\sum_{n} \sum_{r_{mjn}} Q(r_{mjn} \,|\, s_{mj})(\log P(r_{mjn} \,|\, s_{mj}) - \log Q(r_{mjn} \,|\, s_{mj})) \\ &+ \text{const} && (2.16) \end{aligned}$$

where const is a constant that collects all terms that do not explicitly depend on s_{mj}. Solving for $Q^*(s_{mj})$ yields

$$Q^*(s_{mj}) \propto \exp\Big(\log P(s_{mj}) + \sum_{n} \sum_{r_{mjn}} Q(r_{mjn} \,|\, s_{mj})(\log P(r_{mjn} \,|\, s_{mj}) - \log Q(r_{mjn} \,|\, s_{mj}))\Big) \tag{2.17}$$

$$\Rightarrow Q^*(s_{mj}) = \frac{\exp(\log P(s_{mj}) + \sum_n \sum_{r_{mjn}} Q(r_{mjn} \,|\, s_{mj})(\log P(r_{mjn} \,|\, s_{mj}) - \log Q(r_{mjn} \,|\, s_{mj})))}{\sum_{s_{mj}} \exp(\log P(s_{mj}) + \sum_n \sum_{r_{mjn}} Q(r_{mjn} \,|\, s_{mj})(\log P(r_{mjn} \,|\, s_{mj}) - \log Q(r_{mjn} \,|\, s_{mj})))}. \tag{2.18}$$

Note that the update given by Eqn. 2.18 can be efficiently computed, as the operations required to characterize $Q^*(s_{mj})$ can be formulated as matrix and element-wise operations (*eg*, element-wise exponentiation of matrix elements). In particular, such computation is amenable to parallel computing.

Next, we seek $Q^*(r_{mjn} \,|\, s_{mj})$, the expression for $Q(r_{mjn} \,|\, s_{mj})$ that minimizes Eqn. 2.8, keeping all other Q-distributions fixed. We use the same approach as for finding $Q^*(s_{mj})$. Taking the functional derivative of Eqn. 2.14 with respect to $Q(r_{mjn} \,|\, s_{mj})$ and setting it to 0 yields

$$\begin{aligned} 0 = &\log Q^*(r_{mjn} \,|\, s_{mj}) \\ &- \log P(r_{mjn} \,|\, s_{mj}) \\ &- \sum_{g_{mjn}} Q(g_{mjn} \,|\, r_{mjn})(\log P(g_{mjn} \,|\, r_{mjn}) - \log Q(g_{mjn} \,|\, r_{mjn})) \\ &+ \text{const} \end{aligned} \tag{2.19}$$

where const is a constant that collects all terms that do not explicitly depend on $Q^*(r_{mjn} \,|\, s_{mj})$. Solving for $Q^*(r_{mjn} \,|\, s_{mj})$ yields

$$Q^*(r_{mjn} \,|\, s_{mj}) \propto \exp(\log P(r_{mjn} \,|\, s_{mj}) + \sum_{g_{mjn}} Q(g_{mjn} \,|\, r_{mjn})(\log P(g_{mjn} \,|\, r_{mjn}) - \log Q(g_{mjn} \,|\, r_{mjn}))) \tag{2.20}$$

$$\Rightarrow Q^*(r_{mjn} \,|\, s_{mj}) = \frac{\exp(\log P(r_{mjn} \,|\, s_{mj}) + \sum_{g_{mjn}} Q(g_{mjn} \,|\, r_{mjn})(\log P(g_{mjn} \,|\, r_{mjn}) - \log Q(g_{mjn} \,|\, r_{mjn})))}{\sum_{r_{mjn}} \exp(\log P(r_{mjn} \,|\, s_{mj}) + \sum_{g_{mjn}} Q(g_{mjn} \,|\, r_{mjn})(\log P(g_{mjn} \,|\, r_{mjn}) - \log Q(g_{mjn} \,|\, r_{mjn})))} \tag{2.21}$$

Note that just as with the computation of $Q^*(s_{mj})$, the updates in Eqn. 2.21 can be efficiently computed using matrix and element-wise operations.

Next, we seek $Q^*(g_{mjn} \,|\, r_{mjn})$, the expression for $Q(g_{mjn} \,|\, r_{mjn})$ that minimizes Eqn. 2.8, keeping all other Q-distributions fixed. We use the same approach as for finding $Q^*(s_{mj})$. Taking the functional derivative of Eqn. 2.14 with respect to $Q(g_{mjn} \,|\, r_{mjn})$ and setting it to zero yields

$$\begin{aligned} 0 = &\log Q^*(g_{mjn} \,|\, r_{mjn}) \\ &- \log P(g_{mjn} \,|\, r_{mjm}) \\ &+ \sum_{u_{mg}} Q(u_{mg}) \log P(x_{mjn} \,|\, g_{mjn}, u_{mg}) \\ &+ \text{const} \end{aligned} \tag{2.22}$$

where const is a constant that collects all terms that do not explicitly depend on $Q^*(g_{mjn} \,|\, r_{mjn})$. Solving for $Q^*(g_{mjn} \,|\, r_{mjn})$ yields

$$Q^*(g_{mjn}{=}g \,|\, r_{mjn}) \propto \exp\Big(\log P(g_{mjn}{=}g \,|\, r_{mjn}) - \sum_u Q(u_{mg}{=}u) \log P(\vec{x}_{mjn} \,|\, g_{mjn}{=}g, u_{mg}{=}u)\Big) \tag{2.23}$$

$$Q^*(g_{mjn}{=}g \,|\, r_{mjn}) = \frac{\exp\Big(\log P(g_{mjn}{=}g \,|\, r_{mjn}) - \sum_u Q(u_{mg}{=}u) \log P(\vec{x}_{mjn} \,|\, g_{mjn}{=}g, u_{mg}{=}u)\Big)}{\sum_{g'}(\exp\Big(\log P(g_{mjn}{=}g' \,|\, r_{mjn}) - \sum_u Q(u_{mg'}{=}u) \log P(\vec{x}_{mjn} \,|\, g_{mjn}{=}g', u_{mg'}{=}u))\Big)}. \tag{2.24}$$

where to keep the notation uncluttered, we have used the binding notation $\,|\, g_{mjn}{=}g$ to indicate a particular value for g_{mjn}. The updates in Eqn. 2.24 can be computed efficiently using matrix operations.

Finally, we find $Q * (u_{mg})$, the form of $Q(u_{mg})$ that minimizes Eqn. 2.8, keeping all other Q-distributions fixed. We use the same approach as for finding $Q^*(s_{mj})$. Taking the functional derivative of Eqn. 2.14 with respect to $Q(u_{mg})$ and setting it to 0 yields

$$\begin{aligned}0 =& \log Q^*(u_{mg}) \\ &- \log P(u_{mg}) \\ &+ \sum_j \sum_s Q(s_{mj}{=}s) \sum_n \sum_r Q(r_{mjn}{=}r \mid s_{mj}{=}s) \sum_g Q(g_{mjn}{=}g \mid r_{mjn}{=}r) logP(\vec{x}_{mjn} \mid g_{mjn}{=}g, u_{mg}) \\ &+ \text{const}\end{aligned} \tag{2.25}$$

where const is a constant that collects all terms that do not explicitly depend on $Q^*(u_{mg})$. Solving for $Q^*(u_{mg})$ yields

$$Q^*(u_{mg}) \propto \exp(\log P(u_{mg}) + \sum_j \sum_n \sum_g Q(g_{mjn}{=}g) logP(\vec{x}_{mjn} \mid g_{mjn}{=}g, u_{mg})) \tag{2.26}$$

$$\Rightarrow Q^*(u_{mg}){=}\frac{\exp(\log P(u_{mg}) + \sum_j \sum_n \sum_g Q(g_{mjn}{=}g) logP(\vec{x}_{mjn} \mid g_{mjn}{=}g, u_{mg})}{\sum_{u'}(\exp(\log P(u_{mg}{=}u') + \sum_j \sum_n \sum_g Q(g_{mjn}{=}g) logP(\vec{x}_{mjn} \mid g_{mjn}{=}g, u_{mg}{=}u'))} \tag{2.27}$$

where $Q(g_{mjn}{=}g) = \sum_s Q(s_{mj}{=}s) \sum_r Q(r_{mjn}{=}r \mid s_{mj}{=}s) Q(g_{mjn}{=}g \mid r_{mjn}{=}r)$. Note that the update defined by $Q^*(u_{mg})$ can be efficiently computed.

The details of the Q-distribution updates, $Q^*(s_{mj})$, $Q^*(r_{mjn} \mid s_{mj})$, $Q^*(g_{mjn} \mid r_{mjn})$, and $Q^*(u_{mg})$ have now been specified. Note that because of the explicit decompositions we have made in defining the Q-distributions, all of these updates are efficient to compute using matrix operations and element-wise operations.

2.2.3 M-step

With the Q-distributions and associated updates outlined in Sections 2.2.1 and 2.2.2, respectively, we give the details of the model parameter updates to complete our description of the variational EM learning framework. In the language of EM algorithms, we now specify the M-step.

We first explicitly write out the dependence of the complete data log-likelihood on the model parameters by substituting Eqns. 2.1, 2.2, 2.3, 2.4, 2.6 into Eqn. 2.7, yielding

$$\begin{aligned}&\log P(\mathbf{X}, \mathbf{Z} \mid \vec{\theta}, \Phi) = \\ &\sum_{mg} \log \beta_u + \sum_{mj} \log \theta_s \\ &+ \sum_{mjn} \log \pi_{nsr} + \sum_{mjn} \alpha_{mjrg} \\ &+ \sum_{mjn} \sum_h \log \mathcal{N}(x^h_{mjn} \mid \mu^h_u, (\sigma^2)^h_{mg})\end{aligned} \tag{2.28}$$

where, for brevity, we use the notation $u_{mg}{=}u, s_{mj}{=}s, r_{mjn}{=}r, g_{mjn}{=}g$. Substituting Eqn. 2.28 into Eqn.

2.8, the equation for free energy, yields

$$\begin{aligned} F = &\sum_{\mathbf{Z}} Q(\mathbf{Z}) \log Q(\mathbf{Z}) \\ &- \sum_{\mathbf{Z}} \Big(Q(\mathbf{Z}) \big(\sum_{mg} \log \beta_u + \sum_{mj} \log \theta_s \\ &+ \sum_{mjn} \log \pi_{nsr} + \sum_{mjn} \log \alpha_{mjrg} \\ &+ \sum_{mjn} \sum_{h} \log \mathcal{N}(x^h_{mjn} \,|\, \mu^h_u, \vec{(\sigma^2)}^h_{mg}) \big) \Big). \end{aligned} \tag{2.29}$$

We now consider the updates for the model parameters $\Phi = \{\pi, \alpha, \beta, \mu, \sigma\}$. We first consider the update for parameter π_{nsr}: the probability that pixel n of shapelet s takes on the region label r. Setting the derivative of Eqn. 2.29 with respect to π_{nsr} to zero, and using a Lagrange multiplier to enforce the constraint $\sum_r \pi_{nsr} = 1$ (*ie*, $\vec{\pi}_{ns}$ forms a valid probability distribution), we get

$$0 = \sum_{mj} \frac{Q(s_{mjn} = s) Q(r_{mjn} = r \,|\, s_{mjn} = s)}{\pi_{nsr}} + \lambda \tag{2.30}$$

$$\Rightarrow \pi_{nsr} \propto \sum_{mj} Q(s_{mjn} = s) Q(r_{mjn} = r \,|\, s_{mjn} = s) \tag{2.31}$$

$$\Rightarrow \pi_{nsr} = \frac{\sum_{mj} Q(s_{mjn} = s) Q(r_{mjn} = r \,|\, s_{mjn} = s)}{\sum_{mj} Q(s_{mj} = s)} \tag{2.32}$$

where λ is a Lagrange multiplier. The update for π_{nsr} given in Eqn. 2.32 can be thought of as updating the normalized preferences a given pixel n in a given shapelet s has for the region label r, and operates by summing over the evidence in each image and each patch of how well a particular pixel is described by a particular region-shapelet combination. We note that the parameter update given by Eqn. 2.32 can be efficiently computed using matrix operations.

The updates for model parameters β_u, and α_{mjrg} can be derived using the same approach used for the updates for π_{nsr}. The updates for β_u, and α_{mjrg} are:

$$\alpha_{mjrg} = \frac{\sum_s Q(s_{mj}{=}s) \sum_n Q(r_{mjn}{=}r \,|\, s_{mj}{=}s) Q(g_{mjn}{=}g \,|\, r_{mjn}{=}r)}{\sum_s Q(s_{mj}{=}s) \sum_n Q(r_{mjn}{=}r \,|\, s_{mj}{=}s)} \tag{2.33}$$

$$\beta_u = \frac{\sum_{mg} Q(u_{mg}{=}u)}{\sum_{mgu'} Q(u_{mg}{=}u')}. \tag{2.34}$$

The palette parameters, $\vec{\mu}_u$ and $(\sigma^2)^h_{mg}$, are updated in a similar fashion to π_{nsr}, though the derivation does not require the use of a Lagrange multiplier. The updates are given by:

$$\vec{\mu}_u = \frac{\sum_{mjng} Q(u_{mg}{=}u) Q(g_{mjn}{=}g) \vec{x}_{mjn}}{\sum_{mjng} Q(u_{mg}{=}u) Q(g_{mjn}{=}g)} \tag{2.35}$$

$$\vec{(\sigma^2)}^h_{mg} = \frac{\sum_u Q(u_{mg}{=}u) \sum_{jn} Q(g_{mjn}{=}g) (x^h_{mjn} - \mu^h_u)^2}{\sum_u Q(u_{mg} = u) \sum_{jn} Q(g_{mjn}{=}g)}. \tag{2.36}$$

Note that the palette updates resemble the mean and variance updates for learning a mixture of Gaussians, where the responsibilities are given by $Q(u_{mg}{=}u)Q(g_{mjn}{=}g)$ and the datapoints are the pixels.

All the model parameter updates have now be specified, and the description of our variational learning approach is complete.

2.3 Variational Inference

We now describe how we perform variational inference in the shapelet model. Inference on a given *test* image m involves computing the approximate posterior distributions $Q(s_{mj})$, $Q(r_{mjn} \mid s_{mj})$, $Q(g_{mjn} \mid r_{mjn})$, and $Q(u_{mg})$, as well as the image and patch-specific model parameters $\vec{\alpha}_{mjr}$, and $(\vec{\sigma^2})_{mg}$. Note that the model parameters that are shared across images, such as the parameters governing the shapelet library, $\vec{\pi}_{ns}$, the distribution over universal colors, $\vec{\beta}$, and the mean of the universal colours $\vec{\mu}$ are held fixed during inference as they are viewed as parameters that describe the world of images (or just a particular dataset), and so should not vary from image to image. Therefore, variational inference is carried out using the same variational EM algorithm as in Sections 2.2.2 and 2.2.3, except keeping the model parameters $\vec{\pi}_{ns}$, $\vec{\beta}$, and $\vec{\mu}$ fixed.

Chapter 3

Experimental Results

In this section, we describe the experiments performed to analyze the properties and performance of the shapelet model. In particular, we performed experiments using the Caltech101 [13], and Caltech28 [2] datasets to examine object recognition performance, and Caltech28 [2] to examine image reconstruction capabilities. We note that since we have a generative model of images, it is instructive to look at the kinds of images are framework is able to reconstruct, and the properties of those reconstructions. Also, in order to compare more directly with existing stel model variants such as [20], we examine scene recognition performance using the "All I Have Scene" (AIHS) dataset [11]. Although we have motivated our work through an object recognition paradigm, scene recognition poses similar challenges (*eg* variations in scene illumination and viewing angle), and so examining scene recognition capabilities is instructive to understanding the shapelet model.

3.1 Dataset description

We use several datasets to qualitatively evaluate our framework on object recognition, and scene recognition. In this section, we describe the datasets we use to evaluate our shapelet model.

3.1.1 Caltech101

The Caltech101 dataset [13] has recently become a standard benchmark for object recognition algorithms. The dataset is comprised of 101 object classes and one "background" class, with each class containing between 31 and 800 examples. Images are a mix of RGB and gray-level pixel values, and are sized roughly to be $300x200$ pixels. The dataset also comes with annotations indicating a tight bounding box around the object, and object contours. However, in this work, such annotations are unused. For a given image, only the class label information is used.

Figure 3.1 shows a few example images from this dataset. As can be seen from the figure, the each image in the dataset contains exactly one instance of one kind of object, objects are cropped, roughly centre-aligned, and all instances of a class are often in a canonical pose (*eg*, motorbikes are all right-facing). Also note that some classes, such as the *minaret* class (row 12, first and second from the left) contains rotation artifacts. Although the aforementioned properties are undesirable in the sense that they are biases in the dataset that are presumably absent in the real world, we benchmark on the Caltech101 dataset so as to compare with other state-of-the-art algorithms.

3.1.2 Caltech28

The Caltech28 dataset [2] is another dataset used to benchmark object recognition algorithms. This dataset is derived from the Caltech101 dataset by extracting images from the (28) object classes that contain at least 60 examples. We report on Caltech28 results so as to compare with another stel model variant [19].

3.1.3 All I Have Seen

In addition to object recognition, we evaluate our approach on scene recognition. For this task, we use the *All I Have Seen* (AIHS) dataset [11]. This dataset seeks to model the distribution of visual data that a human receives on a day-to-day basis. To this end, the AIHS dataset was created by outfitting a single human with a wearable camera for two weeks, having the camera take a picture every 20 seconds during waking hours, and then manually selecting a subset of the captured images.

The dataset consists of 10 classes of recurrent scenes such as a work office, outside of a house, and a biking trail, with each class containing 30 examples. Images are 48×64 pixels. Note that the class instances are instances of the *same* venue. For example, all work office scenes are instances of *one* particular work office at different times of the day and lighting conditions, and slightly different viewpoints. This is different than other scene recognition datasets, such as the "15 scenes" dataset [12], which combine many different work office venues into a single class.

The AIHS dataset is especially useful to test approaches that factorize shape and appearance since a primary source of variation is lighting conditions. Although there are variations in viewpoints of a particular scene and the exact layout of smaller objects in the scene, these variations are relatively minor compared to the lighting changes. Figure 3.2 shows a few example images from this dataset.

Because of the properties of the dataset, an approach that can effectively factorize shape and appearance should be able to well explain the intra-class variation (lighting changes) with appearance descriptors, and should be able to well explain inter-class variation with shape descriptors.

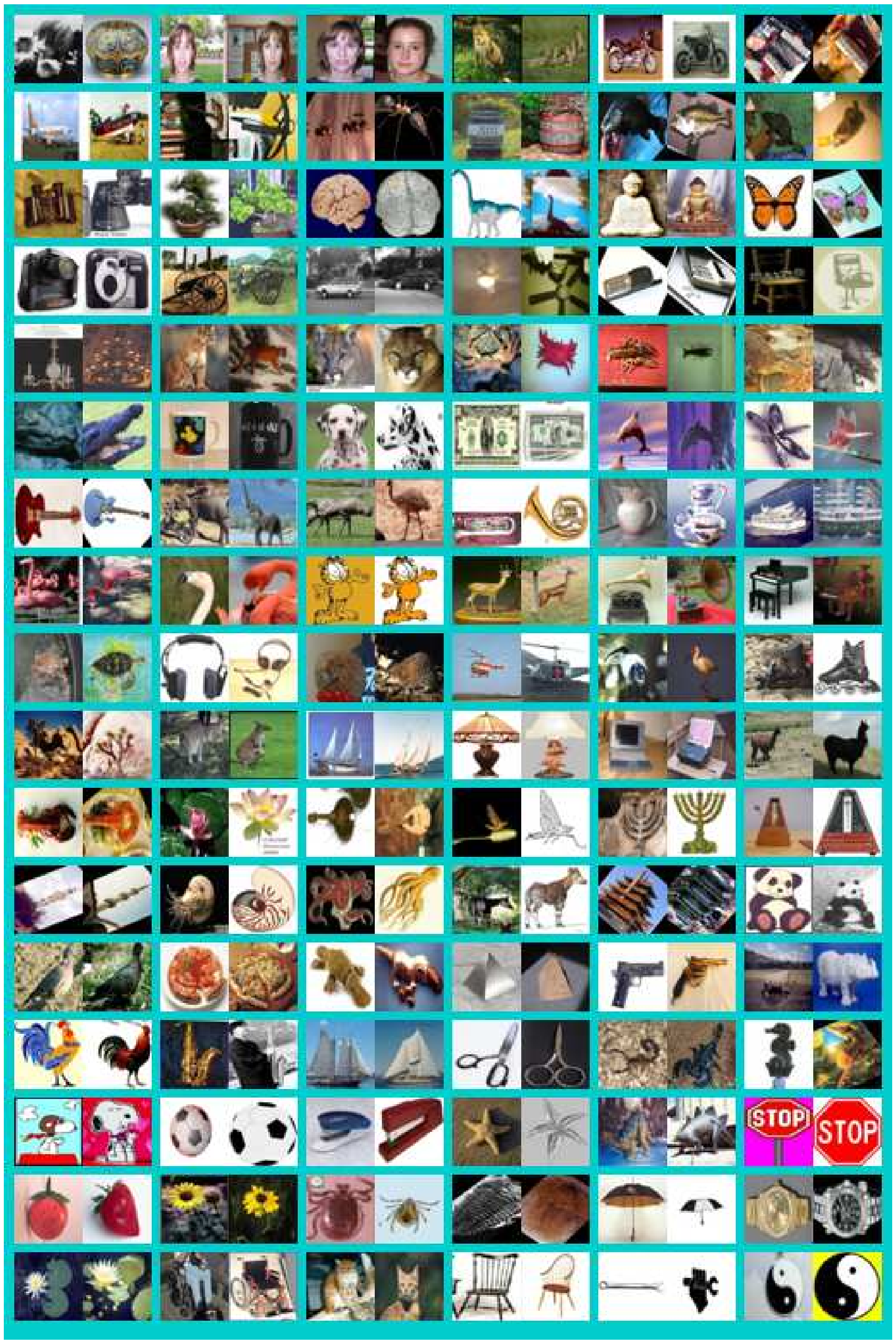

Figure 3.1: Example images from the Caltech101 dataset. Two examples from each of the 101 classes are shown, with both class examples placed side-by-side without space between them. Different classes are spaced part. Note that images are typically cropped, roughly centre-aligned, and class instances often appear in a canonical pose. For example, motorbikes (top row, fourth and third from the right), are right-facing, and are placed in the centre of the image.

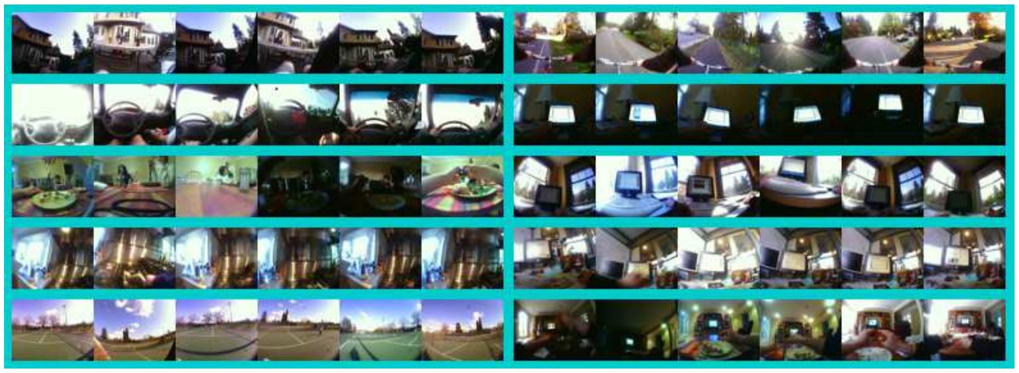

Figure 3.2: Example images from the AIHS dataset. Six examples from each of the 10 classes are shown, with class examples placed side-by-side without space between them. Different classes are spaced part. Note that the instance of a scene class represent a single venue, but with slightly varying viewpoints and layout, and different lighting conditions. For example, the *outside-house* class (top-left row of six images) shows the same house but at slightly different angles and times of day.

3.2 A learned shapelet library

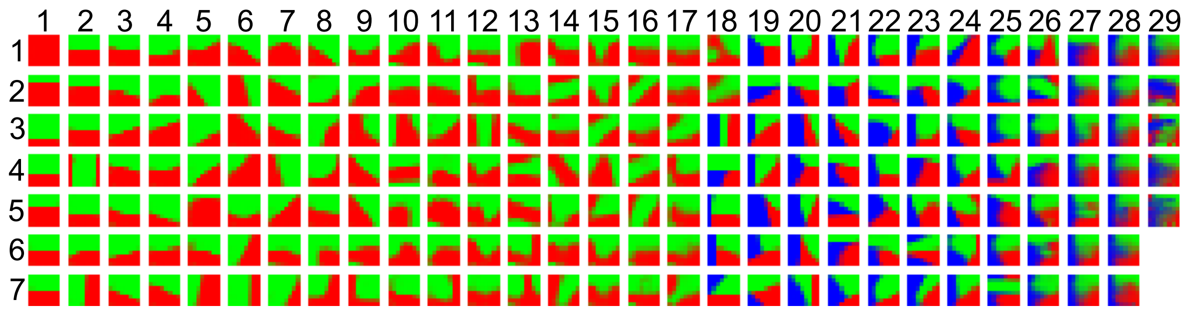

Figure 3.3: A library of 201 shapelets learned from Caltech28 using our shapelet model. For each shapelet, we show the multinomial parameters of the (up to) three regions for each pixel as a linear combination of red, green and blue, where each color represents a multinomial parameter. Prior to learning, the shapelet library was hardcoded to contain one single-region shapelet, 120 two-region shapelets, and 80 three-region shapelets. For visualization purposes, the shapelets are ordered by increasing entropy, and the region colors of red, green, and blue are arbitrary.

We first show the kinds of shapelets (local shapes) our model learns. A set of shapelets learned from the Caltech28 dataset, described in section 3.1.2, is shown in Fig. 3.3. The shapelet library shown in Fig. 3.3 consists of one single-region shapelet, 120 two-region shapelets, and 80 three-region shapelets. We will describe the remainder of the exact parameter settings and implementation used to learn this shapelet library in Section 3.2. As can be seen from the figure, the shapelets learned by our model vary in complexity from simple horizontal and vertical lines (positions (4,1) and (3,12) in Fig. 3.3), to more complex patterns such as quarter circles (positions (4,8) and (1,7)), and multi-line patterns (positions (3,18), (1,24)). Some of the shapelets also resemble Gabor-like filters (positions (2,16), (3,15)), capturing patterns such as lines in different orientations and positions. Note that whereas other coding methods, such as SIFT and Gabor filter responses, would need to have separate filters to account for different combinations of intensity patterns, the shapelet model can make do with fewer filters. In other words, the shapelet model offers a more compact representation. For example, the shapelet position (2,1) can account for a patch with a bright strip above a dark region, but can equally well account for a dark strip above a bright region. Because of this flexibility, the responses to shapelets are more coherent with local shape description than descriptors that do not separate shape and color.

A drawback of the shapelet model approach is that it has no mechanism to encode priors for invariances to image transformations such as scaling and rotation. For example, it may be desirable to instruct a model to learn filter responses that have a small degree of invariance to image transformations, such as small angle rotations, in order to have more compact image descriptors that is robust to noise. In its current form, the shapelet model has no mechanism to support this kind of invariance. However, one may hope that the shapelet model will simply learn about all relevant local patterns if they are prominent in the training set. For example, one may hope that the model learns about lines at all orientations, if such lines do occur in the training set. Unfortunately, as can be seen in Fig. 3.3, this is not the case. While the model learns about lines at particular orientations (positions (2,16), (3,15), (3,12)), the shapelets that represent lines do not densely tile the space of orientations. The same can be seen for learning lines at the same orientation but different thicknesses, which can be viewed as lines at different spatial scales.

For qualitative comparison, we also show the learned shapelets for the patch-based mixture of probabilistic index maps (PB-MoPIM) model; see Section 3.5.3 for a description of this model. Fig. 3.4 show stels learned by PB-MoPIM for a setting where all stels contain $R = 3$ stels. The stels learned largely resemble Gabor-like filters. Additionally, the stel libraries learned by this method generally have higher entropy (and so appear more "blurred") than shapelet libraries. Lastly, stel libraries learned by PB-MoPIM tend to capture simpler shapes, such as lines and curvatures, than shapelet libraries learned by the shapelet model, which tend to capture more complex shapes, such as pie-wedges (position (4,15) in Fig. 3.3).

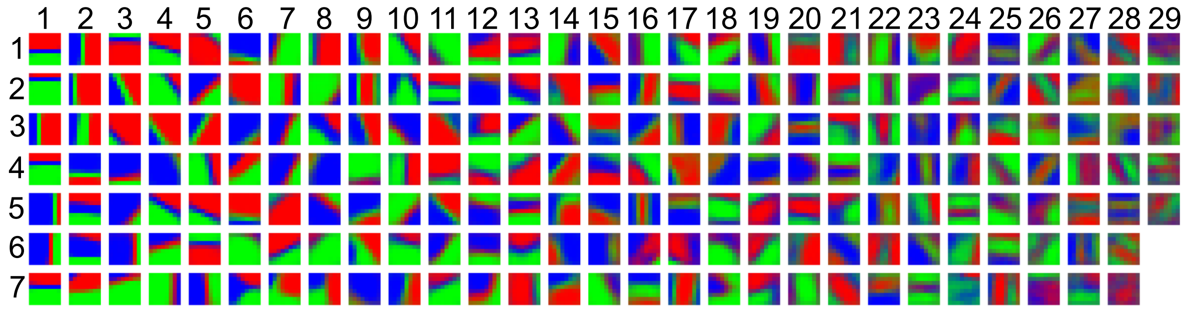

Figure 3.4: A library of 201 probabilistic index maps learned from Caltech28 using the PB-MoPIM model with three regions. For each probabilist index map, we show the multinomial parameters of the three regions for each pixel as a linear combination of red, green and blue, where each color represents a multinomial parameter. For visualization purposes, the shapelets are ordered by increasing entropy, and the region colors of red, green, and blue are arbitrary.

3.3 Shapelet-based patch descriptors

Before we present results on image reconstruction, and object and scene recognition, we describe the image representation used by the shapelet model. Recall that the motivation of the shapelet model is that it factorizes local shape and appearance, and so our image representation is based on quantifying shape and appearance separately.

We first describe how local shape is encoded by the model. Since shapelets represent local shapes, one may hope that given a library of shapelets, local shape in a given image m and patch j can be described by the posterior distribution over shapelets, $P(s_{mj} \mid \vec{x}_{mj})$. However, recall from Section 2.2.1 that computing this posterior distribution is intractable. As a result, we use our approximate shapelet posterior, $Q(s_{mj})$, as our shape-based descriptor. This results in an S-dimensional distribution over shapelets that describes the local shape in an image patch.

Additionally, we can describe image patches according to local appearance features. In the shapelet model, we encode color information. One way to do this, which is *not* done in this work, is to infer, for a given patch in a given image, the distribution over univerisal colour entries. That is, for each patch, one could construct a histogram with U bins and assign each pixel to a single bin according to its most likely universal colour, given by the distribution over the hidden variable $u_{mg_{mjn}}$. To find a pixel's assigned universal color, one could look at its mostly likely global colour, g_{mjn}, and use that as an index to look up its most likely universal color, $u_{mg_{mjn}}$.

There are several problems with the aforementioned approach to describe color information. First, because the universal palettes are specific to our method, using these palettes makes comparison to other algorithms difficult. Ideally, we would like our color descriptor to be approach-agnostic in the sense that most existing algorithms can trivially make use of the same color descriptor. This would allow us to fairly analyze the effectiveness of our shape descriptor, and factorization of shape and color. Second, the model may use the "wrong" universal color in the sense that there is another universal color, if assigned the same pixels, would yield a higher data likelihood. This primarily occurs due to local minima issues in the inference algorithm.

The approach we take to form the patch color descriptor is to form a color histogram. Specifically since our data is described in H color channels, we divide this H-dimensional space evenly into U bins. For example, for RGB space ($H = 3$) and $U = 125$, each of the H color channel is divided evenly into 5 bins, and the bin centres are described by a 3-D vector in RGB space. Then, to form the color descriptor for a patch, we assign pixels to their closest bin centre in this H-dimensional space. Since there are U bins, the patch color descriptor is U-dimensional.

To sum up, we use a factorized description for shape and appearance for a patch. To describe shape, we use the S-dimensional approximate posterior distribution over shapelets, $Q(s_{mj})$, and to describe appearance, we use a U-dimensional color histogram of equally spaced bins.

3.4 Image reconstruction

In this section, we explore the shapelet model's ability to perform image reconstruction. Since we have a generative model, our shapelet model can perform image reconstruction. To perform reconstruction of an image m, we first take a learned shapelet library, which defines the $\vec{\pi}_{ns} \forall n, s$, and using the method outlined in Section 2, we infer an approximate posterior distribution over shapelets $Q(s_{mj} \,|\, \vec{\theta}, \mathbf{X}_{mj}) \forall j$ ($\mathbf{X}_{mj} = \{\vec{x}_{mjn}\}_{n=1}^{N}$ refers to all pixels in a particular patch), the image palette parameters $\vec{\mu}_{mg}$, $(\vec{\sigma}^2)_{mg} \forall g$, and the patch palette parameters $\vec{\alpha}_{mjr} \forall j, r$. We then follow our generative model described in Section 2.1, except using the approximate posterior distribution over shapelets instead of the prior over shapelets given in Eq. 2.5. In addition, instead of sampling from our multinomial distributions, we use the index with the highest probability mass, and instead of sampling from our Gaussian observation models, we use their means. Reconstructions are shown in Figure 3.5 using a shapelet library learned on Caltech28 . The original images were resized to be 100×100, and we performed inference using a patch size of 8×8, a stride of 2[1], and $G = 6$.

Since we use overlapping patches, we also average the appearance of a pixel over all patches that overlap with that pixel. As shown in the image reconstructions, our model is capable of reconstructing images with a variety of properties, such as having very many or very few colors. Additionally, while our model does not typically smooth over object boundaries (due to the typically low-entropy of the learned shapelet library, and using a small stride to perform the image reconstructions), it tends to smooth over areas of fine detail, such as textured regions. This can be seen in the way the "streaking" across in the sky in the original image (left image in Fig. 3.5) is lost in the reconstruction, and in the way the dots on the dog's nose in the original image is smoothed over in the reconstruction. This inability to reconstruct textured regions is to be expected, as modeling co-occurrence of colors in a patch does not lend itself well to explaining texture. Furthermore, the shapelet model assumes that pixels in a patch are independent given the shapelet label, but a hallmark of textured regions is the spatial structure that the pixels in the region exhibit (*ie* pixels values are not independent). Unfortunately, it is non-trivial to modify the shapelet framework to handle such intricate spatial structures that textured regions typically display.

[1]our model, described in Sec. 2, assumes non-overlapping patches, but the patch size and stride used for our experiments violates this assumption. We ignore this violation for computational reasons, and to simplify the inference and learning algorithms.

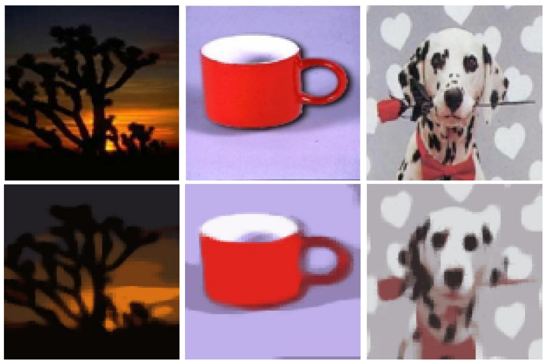

Figure 3.5: Image reconstructions (bottom row) and the original resized 100×100 images (top row). For pictures with a multitude of colors, such as the sunset picture (left), our model finds G colors that captures the dominant colors in the image. Our model can also reconstruct images where there is a relatively small range of colors, such as in the cup picture (middle). Finally, we note that our model tends to smooth regions with fine details, such as the spots on the dog's nose and ears (right).

3.5 Object and scene recognition

To explore the usefulness of the shapelet model for object and scene recognition, we compare it with other competitive methods such as other stel model variants, and a SIFT-based framework. For object recognition, we benchmark using the Caltech28 [2] and Caltech101 [13] datasets, and for scene recognition, we benchmark using the AIHS dataset [11].

We first describe how we form image descriptors are formed from patch descriptors, and how we use these image descriptors to perform object and scene recognition. We then describe the experimental protocol, and then present performance results on Caltech28 , Caltech101 , and AIHS datasets.

3.5.1 Shapelet-based image descriptors

Before forming *image* descriptors for an image, we first form *patch* descriptors for each patch in the image using the approach described in Section 3.3. Recall that we use two types of patch descriptors: shape descriptors and color descriptors, represented by the S-dimensional approximate posterior over a given shapelet library, and a U-dimensional color histogram, respectively. Although we could simply concatenate all patch descriptors together in raster scan order, thus forming a $J \times S$-dimensional shape descriptor, and a $J \times U$-dimensional color descriptor, that approach has severe drawbacks. First, the descriptors would be sensitive to small transformations such as translations. For example, shifting a object in an image over by a few pixels could cause a permutation all the entries in the image descriptor. Second, the descriptors could become very high-dimensional and may cause working with certain kinds of classifiers computationally intractable. For example, if $J = 2000$, $S = 201$, $U = 125$, which as will be discussed later are typical values for these parameters, the shape descriptor would be $2000 \times 201 = 402,000$-dimensional, and the color descriptor would be $2000 \times 125 = 250,000$-dimensional. Such high-dimensional descriptors would make even simpler classifiers like K-nearest-neighbors difficult to use.

Because of the issues with concatenating patch descriptors in raster scan order outlined above, we follow the approach of [12, 24]. Specifically, we spatially pool features using an image pyramid with three levels which divide the image into 1×1, 2×2 and 4×4 non-overlapping blocks, leading to a total of $1 + 4 + 16 = 21$ spatial regions. To do the pooling, we use an *averaging* operator. Spatial pooling helps with increasing robustness to small translations since such translations in the input will not yield a different image descriptor; only translations large enough to cause patches to land in a different image region will affect the pooled feature representation. Also, since there are only 21 spatial regions instead of J, as would be the case if all patch descriptors were concatenated together, the dimensionality of the image descriptors is greatly reduced.

To make features gathered over all levels of the pyramid comparable, we normalize each type of descriptor (shapelet and color) for each of the 21 regions to have unit length. The complete image representation consists of $21 \times S$ descriptors of local shape, and $21 \times U$ descriptors of local color, each of which are normalized to have unit length. This completes the description of forming image descriptors from patch descriptors.

3.5.2 Shapelet-based classifier

We now discuss how to perform object and scene recognition given a set of image descriptors describing the training set and a test image.

Given a shapelet library, learned using the algorithm described in Section 2, we first construct image constructors for the training set using the method described in Section 3.5.1. Following this, we train a one-versus-all SVM classifier [3], setting the soft-margin penalty to $C = 1$ for each class. For the SVM kernel function, we use a convex combination of two kernels- one kernel measures similarity in shape, and the other similarity in color. Specifically, we use the kernel

$$K(\vec{x}^A, \vec{x}^B) = wK_s(\vec{x}_s^A, \vec{x}_s^B) + (1 - w)K_c(\vec{x}_c^A, \vec{x}_c^B) \tag{3.1}$$

where $\vec{x}_s^A \in \mathcal{R}^S$ is the shapelet descriptor and $\vec{x}_c^A \in \mathcal{R}^U$ is the palette (color) descriptor for image A, K_s measures similarity between shapelet descriptors, K_c measures similarity between palette descriptors, and $w \in [0..1]$ is the weighting between the two. w controls the importance of shape similarity versus color similarity; $w > 0.5$ indicates a belief that shape is more discriminative for determining class (object or scene) membership, $w < 0.5$ indicates a belief that color is more discriminative, and $w = 0.5$ indicates a belief that shape and color are equally discriminative.

There are many reasonable kernel functions to use for the similarity functions K_c and K_s, such as dot-product, cosine angle, and Gaussian kernel with an appropriate bandwidth. The intersection kernel, defined as $K(\vec{x}, \vec{y}) = \sum_i^M min(x_i, y_i)$ where M is the dimensionality of both $\vec{x}$ and $\vec{y}$ has been empirically successful in standard baselines [12, 7] and is straightforward to compute. For these reasons, we have adopted the intersection kernel as the similarity measure for both shape (K_s), and color (K_c).

3.5.3 Baselines

We compare our shapelet model with other stel model variants, including the multi-level stel model [19], the Stel Epitome model [20], and a patch-based version of the original stel model (the PIM model [10]) where possible. Recall that the PIM model outlined in [10] operates on entire images, as opposed to patches. In order to provide a fair basis of comparison with our patch-based shapelet framework, we modify the PIM model to operate as a *mixture model* on patches. Specifically, we modify the PIM learning algorithm to learn a PIM mixture model for patches by dividing an image into (possibly overlapping) patches, and treating each patch as a separate image. We learn the parameters of the K PIMs, where K is the number of mixture components, using an EM algorithm resembling that of the algorithm used to learn shapelets, and also infer a distribution over the K PIMs for each patch in a similar fashion to how the shapelet model infers a distribution over K shapelets. For classification, we use the posterior distribution over stels for each patch as the local shape descriptor, and use a color histogram to form the local color descriptor as described in 3.5.1. We then proceed proceed with spatial pooling and SVM classification as outlined in 3.5.2. Note that from the SVM classifier's perspective, the only difference between our method and the PIM method is the shape descriptor. For all experiments involving our patch-based PIM model variant, we use a patch size of 8×8 pixels, a stride of 2 pixels, and a library of 201 stels with each stel having three regions. The size of the library and number of regions per stel were selected so that the library complexity (number of free parameters) is comparable to the complexity of the shapelet model's library. *We will henceforth refer to this PIM model variant as PB-MoPIM for "Patch-Based Mixture of PIMs".*

To compare with the multi-level stel model, we restate the results as published in [19] [2] on available datasets.

We also compare with the state-of-the-art method of using SIFT descriptors with vector quantization, and an SVM classifier stage with the Spatial Pyramid Match kernel (SIFT+SPM) [12] for classification. This method extracts SIFT features and performs vector quantization as outlined in Section 1.2, and employs the same spatial pooling scheme as in the shapelet model, as described in Section 3.5.1. However, unlike the shapelet method, the SIFT+SPM approach uses the spatial pyramid match kernel [7, 12], which is similar to the intersection kernel except matches at finer levels of the image pyramid are up-weighted to reflect "more discriminative" matches.

We use our own implementation of the SIFT+SPM method as described in [12]. For all experiments, we use a patch size of 16×16 pixels, and a stride of 4 pixels to extract SIFT descriptors. We use a visual dictionary size of 201. For computational efficiency, we learn the codebook by selecting a random set of $100,000$ SIFT descriptors from the training set, and perform $K - means$ on this set of descriptors to learn the visual dictionary. In order to make a fair comparison, we also augment the SIFT+SPM method with color descriptors. We form the local color descriptor in the same fashion as for the shapelet model, as described in 3.5.1, and use the similarity metric defined by Eq. 3.1, where we replace the shapelet feature vector by the the descriptor of visual codewords collected over a spatial pyramid.

3.5.4 Caltech101: Object recognition results

For each object class, we use 30 training examples and up to 30 testing examples. We use five different randomly chosen train/test splits to obtain confidence intervals, and the splits are identical across the different methods. In all experiments, we used RGB color images where possible, and for computational reasons, resized all images to 100×100 pixels. This mimics the experimental methodology of [12], except with image resizing. Note that, as noted in Section 3.1.1, some classes contain as few as 31 examples per class, so a class may contain as few as a single test instance, since the other 30 instances are used for training. The implication of this is that the class distributions at train and test time are vastly different. This violates the pervasive assumption in machine learning, from which SVMs originate, that the training and test data come from some common underlying distribution. In view of this, we acknowledge that this experimental methodology is flawed, especially when coupled with an SVM classifier, but we use this methodology regardless so that we may compare fairly with benchmark approaches that have adopted this methodology.

To perform classification, we must set the weighting, w, between the shapelet and color similarities in Eq. 3.1. We report results for the settings $w = \{0.5, 1\}$ which corresponds to classification with an equal weighting of the shape and color descriptors, and classification using only shape descriptors. Although it is possible to cross-validate the setting w on a per-class basis, which corresponds to a belief that the relative importance of shape and color is class-dependent, we have found in practice the performance gain is negligible.

[2] we were unable to run their code on our partitions as the code is unavailable.

Method	Shape only ($w = 1$)	Shape + color ($w = 0.5$)
Shapelets	$59.1\%(0.5\%)$	$\mathbf{63.4\%(1.1\%)}$
PB-MoPIM with $R = 3$	$59.77\%(0.9\%)$	$\mathbf{63.5\%(1.1\%)}$
SIFT+SPM[3]	$58.3\%(0.8\%)$	$61.6\%(1.0\%)$

Table 3.1: Caltech101 classification rates. The standard deviation of the estimated mean classification rate is shown in brackets. Note that our model, when only shape descriptors are used, achieves comparable performance to the baselines methods. However, when color is added, we slightly outperform the other methods.

For the shapelet model, we used 8×8 patches with a stride of 2our model, described in Sec. 2, assumes non-overlapping patches, but the patch size and stride used for our experiments violates this assumption. We ignore this violation for computational reasons, and to simplify the inference and learning algorithms., set $G = 6$ and $U = 125$, used a shapelet library having one single-region shapelet, 120 two-region shapelets, and 80 three-region shapelets, and set $\lambda = 2$. 8×8 patches are large enough to capture interesting local structure in 100×100 images while still being computational to handle. Our results are not sensitive to the settings of G and U; setting $G = [3...8]$ and $U = [64, 125, 216]$ yields no statistical difference in recognition rates.

The parameters for the baseline methods we have implemented, specifically the PB-MoPIM model and the SIFT+SPM framework, are given in Section 3.5.3.

We report classification results for $w = \{0.5, 1\}$ in Table3.1 as the mean of the diagonal of the confusion matrix.

Comparing with the SIFT+SPM approach, it is interesting to note that when ignoring color information and classifying based on shapelet similarity alone ($w = 1$), we achieve comparable recognition rates to the SIFT+SPM approach, but we outperform them when color information is included. This difference in performance is due to our model's explicit factorization of local shape information from local color information. Because of this factorization, the shapelet model can make greater use of color information when it is provided. On the other hand, SIFT descriptors, which are a function of image gradients within a patch, already incorporate a form of color information and so do not gain as much when explicit color information is provided. In other words, color is, in a sense, an orthogonal source of information when combined with our shapelet descriptors, while for the SIFT+SPM approach, gradient statistics of color are already accounted for and so raw color does not constitute an orthogonal source of information.

Comparing with the PB-MoPIM model, we note that our performance is competitive; the performance difference is not statistically significant. Examining the shape descriptor-only performance ($w = 1$), it is surprising that PB-MoPIM achieves similar performance as the shapelet model. The PB-MoPIM model learns mainly simple Gabor-like patterns (see Figure 3.4), while the shapelet model learns both simple and more complex patterns, such as pie-wedges and curvatures (see Figure 3.4), which may be more discriminative patterns for recognizing object classes. Unfortunately, the results for this dataset suggest that the complex patterns that the shapelet model learns does not help in object recognition. There are a few possible reasons for this. First, the complex patterns may be generic patterns that happen to occur for many classes (*eg* many classes contain curvature patterns) and so are not actually discriminative. Also, it may be that the complex patterns are modelling common background clutter, which again would not make them discriminative patterns. Lastly, since there is much more complexity and variation in the kinds of local shapes learned by the shapelet model than by the PB-MoPIM model, more training examples may be needed by the SVM classifier to appropriately characterize the shapelet feature space. To

[3]We do not exactly reproduce the results reported by [12] of 64.6% using 30 training examples due to our image resizing

Method	Descriptor only ($w = 1$)	Descriptor with color ($w = 0.5$)
Shapelets	$74.6\%(0.4\%)$	$\mathbf{83.1\%(0.5\%)}$
PB-MoPIM with $R = 3$	$74.2\%(0.3\%)$	$79.9\%(0.3\%)$
SIFT+SPM	$75.4\%(0.4\%)$	$79.7\%(0.4\%)$

Table 3.2: Caltech28 classification rates. The standard deviation of the estimated mean classification rate is shown in brackets. Note that as with the Caltech101 results, our model, when only shape descriptors are used, achieves comparable performance to the baselines methods. However, when color is added, we slightly outperform the other methods.

investigate this possibility, we examine the effect of training set size on performance in Section 3.5.7. Finally, the PB-MoPIM model and shapelet model appear equally effective at using additional color information ($w = 0.5$).

One question to ask is how useful color alone is for classification. Setting $w = 0$ so that only color is used as the descriptor yields a classification rate of $27.6\%(0.4\%)$.

Finally, we ran our method on the same three train/test partitions the authors of the multi-resolution stel model [19] used to evaluate their method. In this setting, 15 training and 15 test examples are used. Note that these additional trials are *not* included in the set of trials reported in Table 3.1. The authors of [19] report a recognition rate of 58.92%, while we achieve a recognition rate of $57.9\%(0.6\%)$ with a setting of $w = 0.5$ on these partitions, showing that we have competitive performance on this dataset. It is also worth noting that the classifier used in [19] is much more complex than our simple SVM classifier, and so it is unclear whether the multi-resolution stel model's superior performance is due to the model itself, or the classifier used. We also acknowledge that the multi-resolution stel model could make use of local color information to boost performance[4].

Overall, we note that our results are competitive with the standard stel model applied to patches as well as the multi-resolution stel model. Additionally, our framework slightly underperforms against the SIFT+SPM approach when only shape information is used, but slightly outperforms the SIFT+SPM approach when using both shape and color information.

3.5.5 Caltech28: Object recognition results

We use the same train/test methodology as outlined in Section 3.5.4 for the Caltech101 dataset, except we use 10 different train/test partitions to get confidence intervals. We use the same model parameters for the shapelet model, and all baselines methods.

We report classification results in Table 3.2 as the mean of the diagonal of the confusion matrix. As with the Caltech101 results, our method outperforms the other methods whenever color information is available, but when it is not available, the SIFT+SPM approach performs better. Using only local color information (*eg* setting $w = 0$) yields a classification rate of $51.4\%(0.4\%)$.

As with the Caltech101 results, it is interesting to note again that when ignoring color information and classifying based on shapelet similarity alone ($w = 1$), we achieve comparable recognition rates to the SIFT+SPM model approach, but outperform that approach when color information is included. This result again is due to our model's explicit factorization of shape and color.

Comparing with the PB-MoPIM model, we note that our performance is comparable when only shape information is used ($w = 1$). This mirrors the results obtained on the Caltech101 dataset (see Section 3.1). However,

[4]we could not use color histograms to boost the performance of [19] as the source code is unavailable

unlike on the Caltech101 dataset, we outperform PB-MoPIM when color information is available ($w = 0.5$), which is surprising since both models have similar mechanisms to perform the shape-color factorization. One possibility is that since the PB-MoPIM model is able to choose palettes for each patch independently, it may be prone to overfitting. That is, small changes in colors or light intensity in a patch may cause PB-MoPIM to select an entirely different stel (or a different distribution over stels) to explain the local pattern in the patch. Because of this, small appearance variations can greatly affect the way local shape is encoded and in this sense, the effects of color and shape are not "cleanly" separated. On the other hand, the shapelet model must use a small number of colors (G colors) to explain the coloring of the entire image, and so it is not as sensitive to small appearance variations. Enforcing a set of G image colors, and having each patch select from these G colors to explain its own appearance, can then be seen as enforcing kind of regularization of image appearance. Because of this albeit unusual form regularization, the shapelet model may provide a more robust description of local color, which also leads to a more robust description of local shape. Because of these more robust shape descriptions, the shapelet model is better able to separate local color from shape and so, may benefit more from transitioning from using shape-only information to shape + color information than the PB-MoPIM model. Alternatively, this may be an instance of an algorithm (in this case, the shapelet model) being well-suited to the peculiarities of a particular dataset.

The multi-resolution stel model as reported in [19] achieves a recognition rate of 78.1% on this dataset, but their approach may benefit from using a color histogram[5]. Additionally, their performance numbers may change if tested on our train/test splits.

It is instructive to look at the kinds of classification errors that the shapelet model makes when only shape information is available, and when only color information is available. Having a factorized representation for shape and color affords us the luxury to try to address this question. Fig. 3.6 shows the confusion matrix of the shapelet model both with and without color information. First, color information is greatly helpful when a class contains a characteristic set of colors. For example, the *dolphin* class typically contains a lot of blue, and so color information proves to be greatly informative in distinguishing this class whereas shape information did not prove to be totally effective. Note also that some class pairs that are not well disambiguated by shape alone, like *sunflowers* and *lotuses*, are well distinguished when color information is also used. Also, the pair *brain* and *soccer ball* classes are also similarly shaped (both somewhat round) and are often confused when only shape information is used, but can be disambiguated when color information is provided. We note again that for such classes, we receive such significant boost in classification rate by adding color information since the shapelet model factorizes description of local shape from description of local color. However, this gain in classification rate when adding color does not in occur all classes. For example, the *ewer* class has a lower classification rate when color is added. This occurs since in the *ewer* images, the background, which is often a uniform color that is different for each image, takes up a large portion of the image. Because of this, the color histograms capture color information primarily about the uninformative background. To compound this difficulty, the *cup* class images appears on similar kinds of backgrounds as the ewer images, and Fig. 3.6 shows that ewers and cups are confused more often when color information is used. In general, adding color information helps to disambiguate classes that are similarly shaped (*eg sunflower* vs. *lotuses*, *brain* vs. *soccer ball*, but can also cause similarly colored classes to be confused (*eg ewer* vs. *cup*).

[5]we could not use color histograms to boost the performance of [19] as the source code is unavailable

Method	Descriptor only ($w = 1$)	Descriptor with color ($w = 0.5$)
Shapelets	$86.9\%(0.5\%)$	$91.9\%(0.3\%)$
PB-MoPIM with $R = 3$	$87.6\%(0.4\%)$	$92.3\%(0.6\%)$
SIFT+SPM	$93.6\%(0.3\%)$	$\mathbf{94.4\%(0.5\%)}$

Table 3.3: AIHS classification rates. The standard deviation of the estimated mean classification rate is shown in brackets.

3.5.6 AIHS: Scene recognition results

We now report experimental results for scene recognition using the AIHS dataset described in Section 3.1.3. For each scene category, we use 15 training examples and 15 testing examples. We use five different randomly chosen train/test splits to obtain confidence intervals, and the splits are identical across the different methods. In all experiments, we used RGB color images where possible, and kept the image to its original 48×64 pixel resolution. This mimics the experimental methodology of [19].

We report classification results in Table 3.3 as the mean of the diagonal of the confusion matrix. Using only local color information (*eg* setting $w = 0$) yields an extremely high classification rate of $90.1\%(0.4\%)$, and is even competitive with the shapelet and PB-MoPIM models. Since color appears to be so discriminative for this dataset, we believe that this dataset is not suitable for investigating the usefulness of shape-color factorization for scene recognition. However, we benchmark on this method for sake of comparison with another stel-related method [20].

Comparing with the SIFT+SPM approach, it is interesting to note that our method is significantly outperformed by the SIFT+SPM approach. Examining the results of Table 3.3, it can be seen that the SIFT+SPM shape descriptors in particular constitute a much more powerful and robust scene representation than the shapelet model. Although our approach may seem to benefit more from adding color information than the SIFT+SPM model for this task, it is difficult to make such a conclusion since the SIFT+SPM model operates in a much lower test-error regime than the shapelet model, and so performance improvements are inherently much harder to achieve. The superior performance of the SIFT+SPM approach is not surprising as SIFT descriptors have been successfully applied to scene recognition [12, 20] and appears especially suited to this task.

Comparing with the PB-MoPIM model, we note that our performance is comparable and the difference is not statisically significant.

Unfortunately, the multi-resolution stel model as reported in [19] did not benchmark on this dataset and the code is not public, and so we cannot report this method's result on scene recognition.

Overall, the performance of both the shapelet model and the PB-MoPIM model are inferior to the SIFT+SPM approach, and so it seems that the particular way in which the shapelet model and the PB-MoPIM model factorize shape and color is not conducive to scene recognition. This is contrast to object recognition (see Sections 3.1, 3.2), in which our approach appears superior to the SIFT+SPM approach.

3.5.7 Performance with fewer training examples

Recently, there has been a drive in the vision community to the "big data" paradigm, which espouses the use of lots training data, and so it has become important to understand how an algorithm's performance scales with the number of training examples. Recent datasets that attempt to bridge the gap between smaller datasets such as Caltech101 , and "big data" include the *Caltech256* [8] and *Pascal VOC 2007* [6] datasets.

In this section, we investigate the performance of the shapelet model and other benchmarks as the number of training examples is varied on the Caltech28 , Caltech101 , and the AIHS datasets described in Sections 3.1.1-3.1.3.

We first examine performance as a function of training examples on the Caltech101 dataset. We use a similar train/test methodology as outline in Section 3.1.1, except we train using $[5, 10, 15, 30]$ examples per class while keeping the number of test examples per class at 30.

The graph of performance is given in Figure 3.7.

The first trend to note is that for all methods, performance appears to increase roughly as the $\log$ of the number of examples per training class. This trend of logarithmically increasing performance with number of training examples is characteristic of many recognition algorithms [7, 23, 26].

The second trend to note is that for all number of training examples per class tested, the shapelet model performs comparably to the SIFT+SPM method when only shape information is used, and outperforms it when when color information is added. This suggests that the shapelet model's factorization of shape and color is effective for any number of training examples used, as opposed to only in a "small data" or "big data" limit. Also, the PB-MoPIM model and the shapelet model are comparable when using shape-only information and shape + color information for all number of training examples per class tested.

The final trend to note is that adding color information benefits all methods, even the SIFT+SPM method which does not explicitly factorize shape and color. This suggests that a trivial way for any method to gain performance on this dataset- simply append a color histogram as an additional patch/image feature. However, the usefulness of color as a patch/image feature in general is still an open question. Adding color information on the Caltech101 dataset may only boost performance due to dataset bias, as described in Section 3.1.1.

Next, we examine performance as a function of training examples on the Caltech28 dataset. The graph of performance is given in Figure 3.8.

First, as with the Caltech101 dataset results, the increase in performance is logarithmic in the number of training examples per class. Similarily, the shapelet model again outperforms the SIFT+SPM when shape+color information is used, but performs comparably when only shape information is used. Surprisingly, unlike the Caltech101 results, the shapelet model appears to outperform the PB-MoPIM model when both shape and color information is used for all number of training examples tested. As noted in 3.2, this is a surprising result and it appears from Figure 3.8 that the shapelet model's superior performance reported in Section 3.2 was not a fluke for the number of training examples tested. The trend in Figure 3.8 shows that the shapelet model outperforms the PIPB-MoPIMM model on all number of training examples tested, and suggests that the difference in performance may further increase if the number of training examples per class was increased. Finally, adding color information appears to help all methods on this dataset, as was the case for the Caltech101 dataset.

Lastly, we examine performance as a function of training examples on the AIHS dataset. The graph of performance is given in Figure 3.9.

The performance curves for the AIHS dataset show similar trends as the curves for the Caltech101 and Caltech28 datasets. The primary difference, however, is that the SIFT+SPM approach outperforms both the shapelet model and the PB-MoPIM model for all number of training exampels per class tested. This suggests that the SIFT+SPM approach is generally superior to the shapelet model, regardless of how much training data is provided.

	Faces_easy	Leopards	Motorbikes	bonsai	brain	butterfly	cougar_face	crab	cup	dalmatian	dolphin	elephant	euphonium	ewer	ferry	flamingo	grand_piano	joshua_tree	kangaroo	laptop	lotus	schooner	soccer_ball	starfish	stop_sign	sunflower	watch	yin_yang
Faces_easy	**98**									1													2					
Leopards		**70**		1		1	1	1	1	3	3	2		1	1			1	7		1		1	7	1	1		
Motorbikes			**96**		1	1			1				1						1		2						1	1
bonsai		1		**73**	4		1	1	1		1	1		1	3	2		1	4		2	1			3	3		
brain				1	**86**	1	3	1		1		1	2					2					5					
butterfly		1	1		1	**71**	1	4			2	1	1	1		1				1	3	1	1	5		2		
cougar_face		2	1			2	**68**	3	1	10	1	2	2	2	1			1	5		4		1	1			1	
crab		2				4	4	**70**		3	4		2	1		2		2	1		2			6	1	1	1	
cup					1	1			**72**	2	2	1	2	6	1	1	2	1		4	1		1	1	1		2	1
dalmatian		5		1	1	1	7	4	2	**63**	1	3				2		3	2		3			2	2	2	2	2
dolphin		1				3	1	4	1		**44**	1	1	1	2	5	1	1	3	1	2	1	1	4	1	1	1	1
elephant	1	2		2	1	1	2	1	3	4	4	**72**	1		2	3			3		1		1	2	2	4	1	
euphonium								1		1	1		**78**		1	1		1			2						1	2
ewer		2		2			1	1	4		1			**80**		1	1	1				1	1	1	1	1	1	
ferry		4		6					3		4		5	1	**76**	6	1	3	1	1	1	6			1		2	
flamingo		1		1		1	1	1	1	2	8	4	1	1	5	**63**		2	1		3	3	1	3	2	2	3	1
grand_piano								1	1		3		1	1	2	1	**91**									2		
joshua_tree		2		2			1			3			1		1	2		**74**	4		3	1		1	1	1		
kangaroo		5		2			2		1	1	5	6		1		1		2	**66**		3		1	3		1	1	1
laptop					1				1		3		1		1		1			**92**								
lotus		1		1		5	3	1	1	1	2	1		1		2		2	1		**47**		2	2		3	1	1
schooner		1							1		1	1		2	1	2						**83**						
soccer_ball				1	1	1		2	1		2			1							3	1	**73**	2			1	3
starfish		1		2		1	1	1		3	6	1				3		2	1		3	1	3	**57**	1	3		
stop_sign				1		1	1		1							1	1	1							**83**		1	
sunflower				1		1	1	1			1				1	1			1		13			3		**72**		
watch					1	4	1	1	1	1				1				1			1	1	1				**81**	1
yin_yang					1																	1	3					**86**

(a) Confusion matrix using only shapelet information

	Faces_easy	Leopards	Motorbikes	bonsai	brain	butterfly	cougar_face	crab	cup	dalmatian	dolphin	elephant	euphonium	ewer	ferry	flamingo	grand_piano	joshua_tree	kangaroo	laptop	lotus	schooner	soccer_ball	starfish	stop_sign	sunflower	watch	yin_yang
Faces_easy	**99**																											
Leopards		**87**					1	1				2				1			4					2				
Motorbikes			**95**			1			1										1			1					1	1
bonsai		2		**82**	3		1	1	1			2			1	3		2	1					1	1		1	
brain					**86**	1	1	2										1					2	1			1	
butterfly						**76**	1	3		2	1	1		2							4			2		3		
cougar_face				1		1	**81**	3	2	4		2		1					6					1				
crab			2		2	2	1	**78**	1	2	4		6			1	1							5	1			
cup					1	2			**70**	1	1		2	10	1		2	1		2			3	2			3	
dalmatian				1	1	3	4	2	4	**84**	1	4				1		1	2		2		3	1	1			
dolphin		1			1			1	1		**77**	1		2	2	2		1	1		2		2	5	1	1		1
elephant			1	3			2	1	2	2	2	**78**	1		1	1			1				1	1	2	2	1	1
euphonium				1									**85**		1		1			1	1						1	1
ewer				1					3		2			**75**		1	1	1	1	1		2						
ferry				2					4		1		3	1	**90**	3		1				5					2	
flamingo		2		1	1	2	1	1	1	1	4	1	1	2		**75**		1	3		4	1		7	3	1	1	
grand_piano									1			1		1		1	**95**									1		
joshua_tree		1		3								1	1					**84**	2		4	1	1			1		
kangaroo		4					2	2				4		1		2		1	**73**		1			4				
laptop					1						1			1						**96**							1	
lotus				2		4	1	1						1		2					**71**		3	2	1	2	2	
schooner									3		1	1		1	1							**90**	2					
soccer_ball									1	1	2			2					1				**78**	2			1	1
starfish			1				1	1		1	1	1				4		1	2		5		3	**65**		1		
stop_sign								1	3												1				**90**			1
sunflower						2						1									2					**89**		
watch				1	1	3	2		1	2								1									**84**	
yin_yang						1								1									2				1	**94**

(b) Confusion matrix using shapelet and color information with $w = 0.5$.

Figure 3.6: Confusion matrices for Caltech28 . Entry (i, j) is the percentage of the time class i was classified as class j, averaged over 10 trials. For pairs of classes not disambiguated by local shape alone, such as the lotus and sunflower classes, and classes where color is highly informative, such as the dolphin class (abundance of blue), adding color information significantly improves performance. However, adding color information hurts performance for a few classes, such as the ewer class. Performance increases are in green circles, and decreases are in red rectangles.

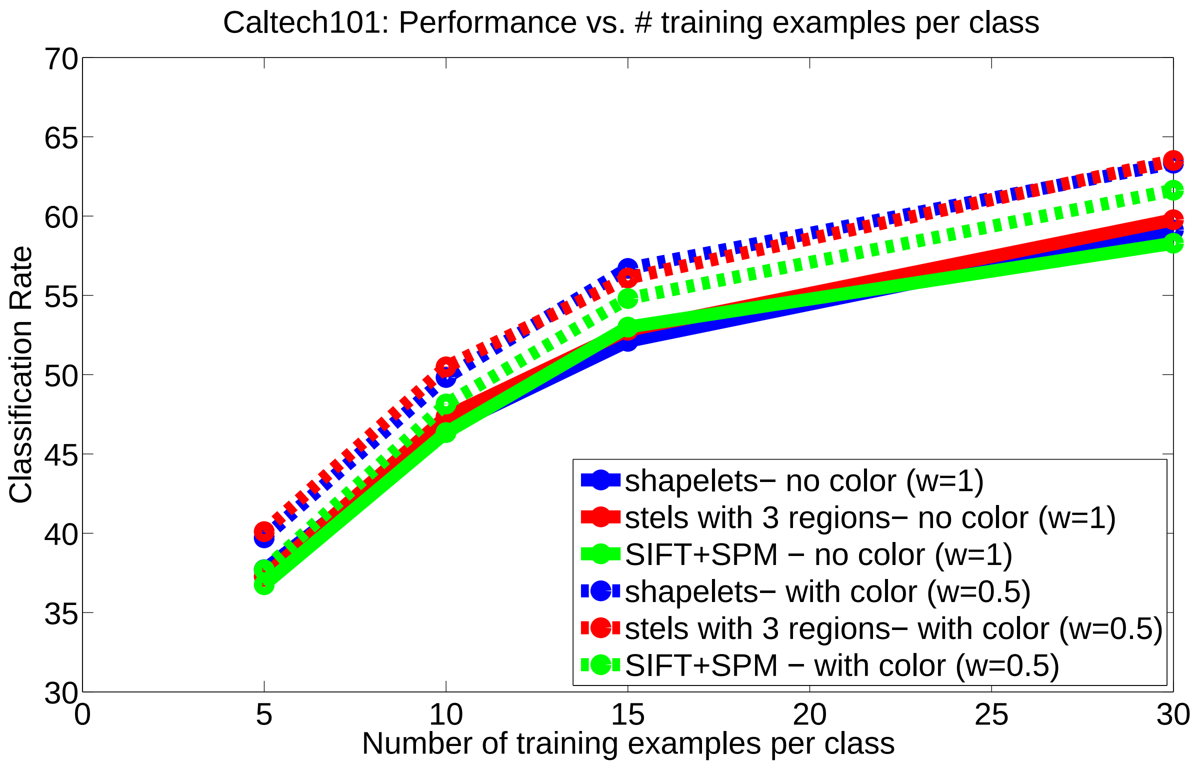

Figure 3.7: Training performance vs. number of training examples per class on the Caltech101 dataset. Note that performance for all methods increases roughly logarithmically, the shapelet model with color outperforms the SIFT+SPM when color information is added for all numbers of training examples tested, and adding color information helps all methods. Also, the shapelet model performs similarily to the PB-MoPIM model on this dataset.

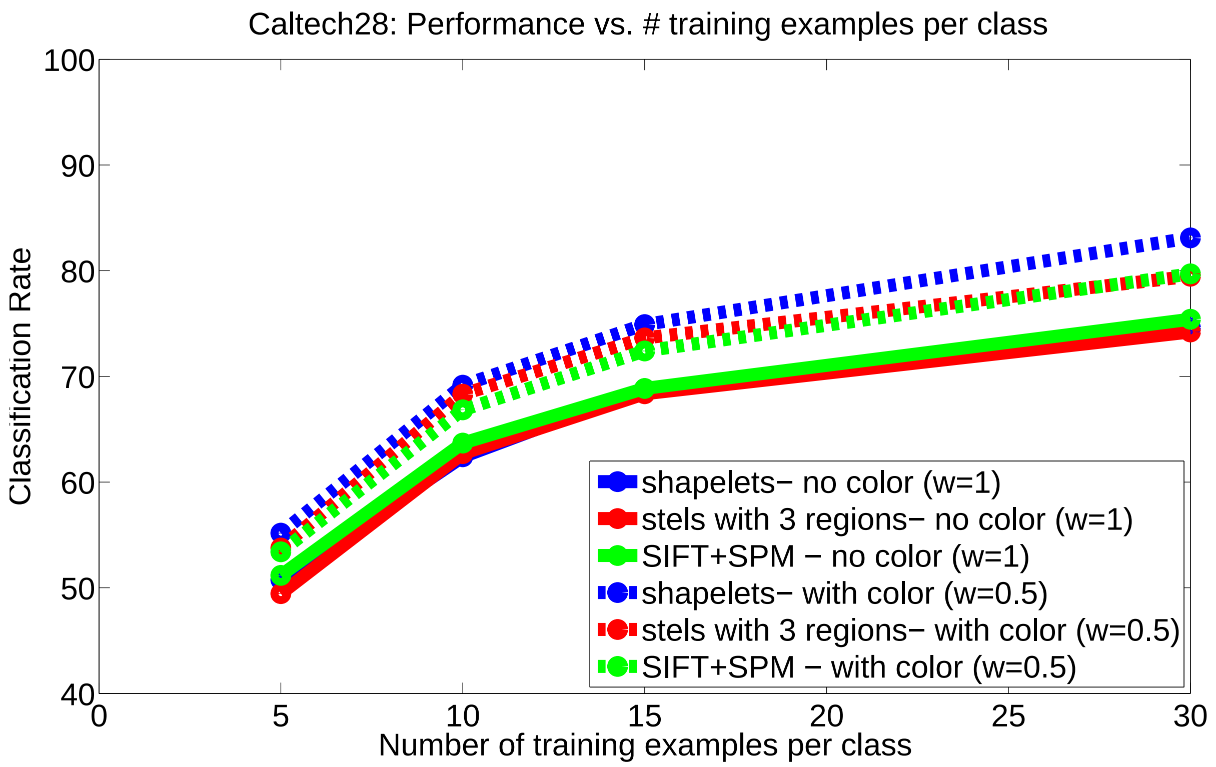

Figure 3.8: Training performance vs. number of training examples per class on the Caltech28 dataset. Note that performance for all methods increases roughly logarithmically, the shapelet model with color outperforms both the SIFT+SPM and PB-MoPIM models when color information is added for all numbers of training examples tested, and adding color information helps all methods.

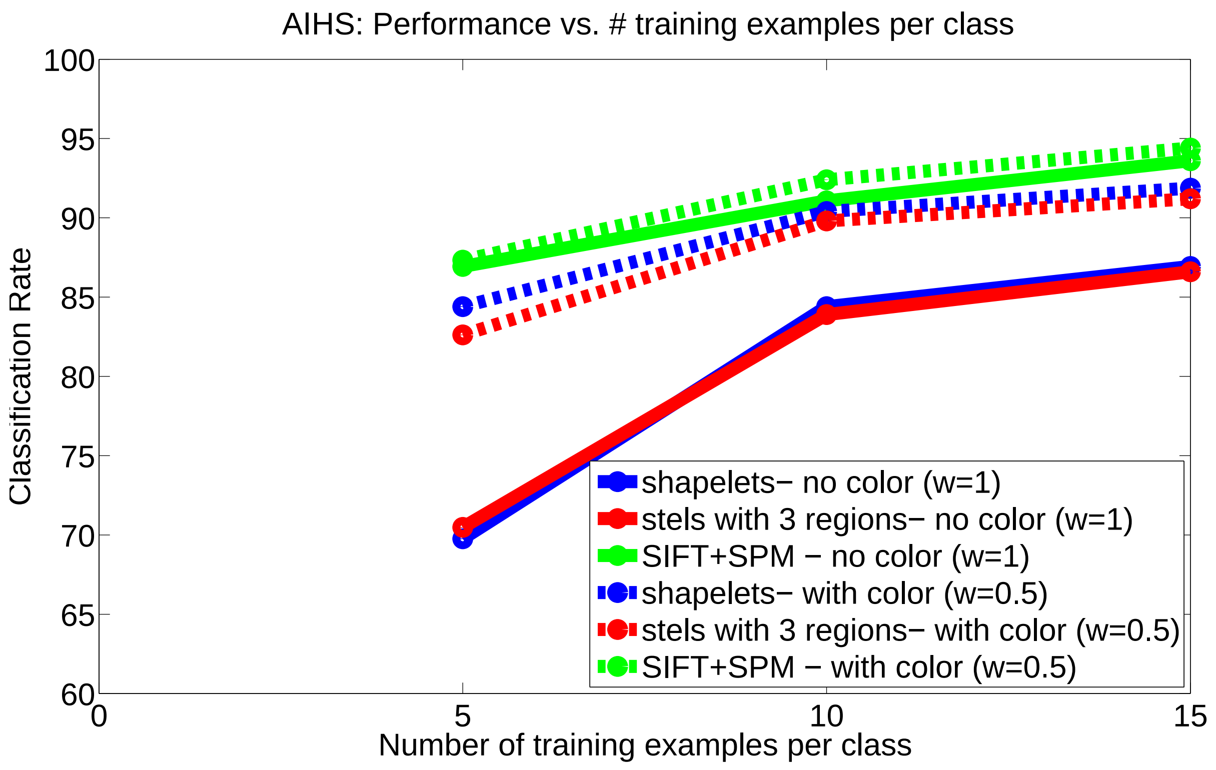

Figure 3.9: Training performance vs. number of training examples per class on the AIHS dataset. Note that performance for all methods increases roughly logarithmically, the SIFT+SPM approach outperforms both the shapelet and PB-MoPIM models when color information is added for all numbers of training examples tested, and adding color information helps all methods.

Chapter 4

Conclusion

In this work, we present an extension of the stel model [10] to a patch-based framework, which we call the 'shapelet model'. We introduce the notion of hierarchical palettes for describing the coloring of an image set, images, and patches in an image. We demonstrate that our framework factorizes local shape from local color in the form of shapelets and palettes, respectively. With such a factorization, object classification and scene recognition can be performed using descriptors encoding shape and color information separately. We illustrate the kinds of local shapes the shapelet model tends to learn, and show that these shapes tend to capture richer structures than a patch-based mixture of PIMs. Also, we show that our model is competitive on the object recognition datasets of Caltech28 , Caltech101 , and the scene recognition dataset AIHS against several baselines.

Bibliography

[1] Matthew Brown and David G. Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73, 2007.

[2] L. Cao and L. Fei-Fei. Spatially coherent latent topic model for concurrent object segmentation and classification. In *ICCV*, 2007.

[3] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[4] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR (1)*, pages 886–893, 2005.

[5] Seyed Mohammadali Eslami and Christopher Williams. Factored shapes and appearances for parts-based object understanding. In *BMVC*, 2011.

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[7] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *In ICCV*, pages 1458–1465, 2005.

[8] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007.

[9] N. Jojic, A. Perina, M. Cristani, V. Murino, and B. J. Frey. Stel component analysis: Modeling spatial correlations in image class structure. In *CVPR*, pages 2044–2051, 2009.

[10] Nebojsa Jojic and Yaron Caspi. Capturing image structure with probabilistic index maps. In *CVPR*, 2004.

[11] Nebojsa Jojic, Alessandro Perina, and Vittorio Murino. Structural epitome: a way to summarize one's visual experience. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1027–1035. 2010.

[12] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[13] Fei-Fei Li, Robert Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1), 2007.

[14] David Lowe. Object recognition from local scale-invariant features. In *Proceedings of ICCV*, 1999.

[15] D. Marr. *Vision: A computational investigation into human representation and processing of visual information*. W. H. Freeman and Company, San Franciso, 1982.

[16] Radford Neal and Geoffrey E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1998.

[17] Aude Oliva and Antonio Torralba. Building the gist of a scene: the role of global image features in recognition. In *Progress in Brain Research*, 2006.

[18] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, June 1996.

[19] Alessandro Perina, Nebojsa Jojic, Umberto Castellani, Marco Cristani, and Vittorio Murino. Object recognition with hierarchical stel models. In *ECCV (6)*, 2010.

[20] Alessandro Perina, Nebojsa Jojic, and Vittorio Murino. Structural epitome: a way to summarize one's visual input. In *NIPS*, 2010.

[21] MarcAurelio Ranzato and Geoffrey E. Hinton. Modeling pixel means and covariances using factorized third-order Boltzmann machines. *JMLR*, 7:2369–2397, 2003.

[22] Eero P. Simoncelli and William T. Freeman. The steerable pyramid: A flexible architecture for multi-scale derivative computation. In *Proc 2nd IEEE Int'l Conf on Image Proc*, volume III, pages 444–447, Washington, DC, Oct 23-26 1995. IEEE Sig Proc Society.

[23] Gang Wang and Ye Zhang Li Fei-fei. Using dependent regions for object categorization in a generative framework. In *CVPR*, pages 1597–1604. IEEE Computer Society, 2006.

[24] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

[25] Matthew D. Zeiler, Graham W. Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *ICCV*, 2011.

[26] Hao Zhang, Alexander C. Berg, Michael Maire, and Jitendra Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *In CVPR*, pages 2126–2136, 2006.

[27] Huiyu Zhou, Yuan Yuan, and Chunmei Shi. Object tracking using sift features and mean shift. *Computer Vision and Image Understanding*, 113(3):345–352, 2009.